DREAMWEAVER MX
Gli strumenti disponibili per la gestione di un sito

COLDFUSION MX
Introduzione all'ambiente e collegamento a database

Supplemento a ioProgrammo n.66

Programmare il WEB

PHP
Un servizio per l'invio di cartoline virtuali

FLASH MX
Creare una lavagna interattiva con ActionScript

Master in ASP
Sezione protetta del sito: le funzioni avanzate

Una Rete a COLORI
Utilizzare grafica e interattività per lasciare il segno

ASP.NET & Web Matrix
Un'applicazione Web per effettuare sondaggi on-line

PENSARE COME UN HACKER: IL CONCETTO DI ROBUSTEZZA

Programmare il WEB

Web Design

Tutorial

Lato Server

Sicurezza

PASSI AVANTI?

Il passo avanti che l'uomo ha compiuto con la diffusione di Internet è stato grande. Tralasciando i comuni vantaggi (e le altrettanto comuni seccature), vorrei soffermarmi su un particolare non abbastanza dibattuto: il ritorno alla parola scritta. Quando ormai la televisione sembrava aver vinto la battaglia della comunicazione, e quando la "civiltà delle immagini" sembrava aver trionfato sul silenzio sbigottito e attonito degli spettatori-cittadini, l'arrivo di Internet ha di colpo fatto saltare questa stagnante situazione. Si leggevano sempre meno libri e giornali, mentre la corrispondenza epistolare sembrava soppiantata dal più comodo e meno impegnativo telefono... Beh, tutto questo è stato letteralmente sovvertito da Internet, che ha fatto riscoprire a tutti, e ai più giovani in particolare, la bellezza e l'utilità del leggere e dello scrivere. Molti hanno contestato l'intrinseca leggerezza della parola che compare sul Web. Devo confessare che anch'io ho l'impressione che quelle parole abbiano una forza ed una consistenza minore rispetto a quelle stampate su carta: diciamo pure che il detto "scripta manent" risulta meno plausibile quando riferito a parole composte di fosfori e cristalli così evanescenti... e non nascondo che, come molti, se ho testi particolarmente importanti da leggere o "studiare", sento la necessità di stamparli. Anche le lettere inviate via mail sembrano obbligare ad una immediatezza e ad una forma di "spontaneità" che va spesso a detrimento della "qualità" della lingua. Ciononostante, resta innegabile un rinnovato interesse verso la parola scritta, con tutto quello che ne consegue in termini di diffusione della conoscenza. A tutto questo si oppone ora una minaccia: la banda larga. Fino ad ora, la ristrettezza di banda ed il costo della stessa hanno fatto sì che l'informazione fosse veicolata prevalentemente in forma testuale. Con l'incremento della banda disponibile, saremo facili profeti nel prevedere che molti degli attuali navigatori sposteranno la loro attenzione dalla pagina Web a forme diverse di intrattenimento come video e film. Il rischio è insomma un nuovo medioevo della parola, un rapido abbandono della parola scritta appena riconquistata. Ovviamente non siamo qui ad augurarci un rallentamento nella diffusione della banda larga, ma ci permettiamo di sollevare un piccolo dubbio: saremo in grado di utilizzare questa tecnologia senza esserne travolti?

Raffaele del Monaco

Programmare il WEB

Supplemento ad ioProgrammo
Anno VII - N.ro 2 (66) - Febbraio 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 - Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo

Dir. Editoriale Massimo Sesti
Dir. Responsabile Romina Sesti
Product Manager Antonio Meduri
Editor Gianfranco Forlino
Redazione Raffaele del Monaco, Antonio Pasqua
Collaboratori M. Battista, A. Cangiano, P. Capitani, C. Giustozzi, F. Mestrone, G. Uboldi
Segreteria di Redazione Veronica Longo

REALIZZAZIONE GRAFICA CROMATIKA Srl
C.da Lecco, zona ind. - 87030 Rende (CS)
Tel. 0984 8319 - Fax 0984 8319225
Coord. grafico: Paolo Cristiano
Coord. tecnico: Giancarlo Sicilia
Impaginazione elettronica:
Aurelio Monaco, Ferdinando Gatto

PUBBLICITÀ Edizioni Master S.r.l.
Responsabile Vendite Ernesto Redaelli
Agenti Vendita Elisabetta Februo, Serenella Scarpa, Cornelio Morari

Segreteria Ufficio Vendite: Daisy Zonato
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321612 - Fax 02 8321764
e-mail: advertising@edmaster.it

EDITORE Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321482 - Fax 02 8321699
Sede di Cosenza: C.da Lecco, zona ind. - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti
Responsabile Amministraz. e Finanza: Benedetto Celsa
Produzione e Logistica: Michele Carere, Eugenio Vocaturo
Diffusione: Alessandra Cervello
Marketing: Giuseppina Bruno, Leonardo Petrone, Antonio Meduri

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel.02 8321482
@ e-mail: servizioabbonati@edmaster.it

Stampa: Elcograf Industria Grafica (LC)
Stampa CD-Rom: Disctronics Italia (MI)
Distribuzione per l'Italia: Parrini & C S.p.A. - Roma

Finito di stampare nel mese di Gennaio 2003

Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, ioProgrammo, Linux Magazine, Softline Software World, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, Il CD-Rom di Idea Web, I Corsi di Win Magazine, Le Collection.

Dreamweaver MX: organizzare il proprio lavoro

Dopo aver visto le caratteristiche principali per la creazione delle pagine HTML con l'ultima versione di Dreamweaver, diamo un'occhiata alle funzioni di gestione di un sito: sfruttiamo la capacità di integrazione con un server e di verifica dei collegamenti tra le pagine, nonché la possibilità di creare asset riutilizzabili per i nostri progetti web.

Nel numero precedente ci siamo occupati delle caratteristiche principali per la creazione di pagine HTML che Dreamweaver mette a disposizione del designer web. In realtà però il compito del prodotto non si esaurisce nell'aiuto per la creazione di codice HTML, ma va ben oltre, permettendo all'utente di lavorare su progetti di respiro più ampio che includono tutte le risorse necessarie alla creazione di un sito (quindi, numerose pagine HTML linkate tra loro, i vari GIF e JPEG associati, applet, e risorse varie) e che potete gestire in maniera molto efficiente grazie ai tool messi a disposizione da Dreamweaver. Inoltre i progetti possono essere legati ad un piattaforma server specifica (a scelta tra ASP, PHP, JSP, ColdFusion, .NET), il che vi permette in un secondo tempo di sfruttare un ventaglio di opzioni di generazione di codice dinamico atte alla creazione di applicazioni web vere e proprie, piuttosto che semplici siti! Inoltre, Dreamweaver si occupa anche della pubblicazione dei file sul server, tenendo a memoria le impostazioni per l'accesso FTP, HTTP, o RDS, oltre ad eventuali note di design che potreste voler associare ad alcuni file, rendendo trasparente l'upload e download dei file da e verso il PC locale. Vedremo insomma che di fatto, una volta inizializzato un sito inteso come progetto Dreamweaver, avrete la possibilità di sfruttare molte funzionalità del prodotto che hanno a che fare con integrazione, versioning e pubblicazione dei file, oltre agli strumenti per aumentare la produttività e l'organizzazione del proprio materiale.

Fig. 1- Il pannello *Site* nel gruppo *Files*

IL SITO COME PROGETTO DI DREAMWEAVER

Per poter accedere alle meravigliose funzionalità promesse nell'introduzione delle righe precedenti, è necessario lavorare su un progetto completo piuttosto che una singola pagina. In Dreamweaver i progetti si chiamano – giustamente – "siti" e per crearne uno si hanno diverse possibilità: dal menu **Site / New Site...**, oppure dal pannello **Site** (lo vedete in **Figura 1**) selezionando le stesse voci di prima dal menu privato del pannello, oppure cliccate sul sottomenu **Edit Sites...** sempre dal menu **Site** e poi selezionate il pulsante **New...**

In un modo o nell'altro, alla fine vi trovate davanti la finestra di creazione di un nuovo sito. Rispetto alla versione 4 del prodotto, qui avete la possibilità di farvi guidare da un wizard nella creazione del progetto, in modo che vi vengono poste delle domande semplici relative alle caratteristiche del vostro sito, del server che lo ospiterà e di come accedere a tale server, per settare così automaticamente le varie opzioni. L'alternativa è quella vecchio stile delle finestre di proprietà, dove avete a disposizione tutto relativamente al sito, ma ovviamente è richiesta un po' più esperienza e una conoscenza più approfondita del prodotto. Entrambe le possibilità, comunque, sono accessibili dalla medesima finestra, è solo una questione di tab (**Basic** o **Advanced**, di cui avete un esempio rispettivamente nelle **Figure 2** e **3**).

Ma vediamo quali sono le scelte che si possono fare nel creare un sito con Dreamweaver. Prenderò come riferimento di base la finestra avanzata, in quanto è quella

Fig. 2- Il wizard per la creazione di un nuovo sito

più completa: vi accorgerete così che tutte le informazioni richieste con il wizard saranno comunque spiegate nella sezione che segue.
Sicuramente due dei parametri più importanti sono il nome (che serve solo da identificativo) e la directory di lavoro (dove sarà salvato tutto il materiale relativo al progetto). È possibile inoltre specificare una directory di immagini di default, dove Dreamweaver copierà eventuali file GIF o JPEG trascinati sui vostri documenti da qualunque punto del vostro computer mentre lavorate su pagine HTML. Esiste poi la possibilità di chiedere a Dreamweaver di gestire una cache locale di informazioni sul sito che velocizza il lavoro e che si rende necessaria qualora vogliate utilizzare gli **asset** (di cui parleremo in seguito).
Continuerei il discorso adesso con la scelta del server. Dreamweaver vi mette a disposizione due blocchi di impostazioni separati: da un lato, configurate la locazione finale delle pagine una volta che sono pronte per la produzione (il web server, insomma), dall'altro indicate a Dreamweaver quale eventuale application server utilizzate di modo che Dreamweaver possa interfacciarsi con esso per aiutarvi a creare e testare pagine dinamiche lato server (il termine usato nel prodotto è **testing server**). Nel primo caso dovete dare al prodotto le informazioni necessarie a connettersi al server per scaricare/caricare le varie risorse web. Avete a disposizione una scelta abbastanza vasta di metodi d'accesso, quali **FTP, WebDAV, RDS, SourceSafe…** In questa sede potete anche specificare se volete che i file vengano rispediti al server ad ogni salvataggio e se volete abilitare il check-in/check-out dei file (niente a che vedere con gli aeroporti!): quest'ultima scelta vi offre un semplice, ma utile meccanismo per lavorare in team. Se qualcuno lavora su un file, viene fatto il check-out del file stesso, che è una specie di lock tramite il quale si impedisce ad altri utenti di modificare il contenuto del file fino a che non venga fatto nuovamente il check-in. Il tutto è gestito trasparentemente da Dreamweaver, il quale vi dà anche la possibilità di segnalare il vostro nome e l'email, di modo che questi dati siano associati a tutti i check-out che fate, cosicché chi tenta di lavorare su un file bloccato sa almeno chi ci sta lavorando e come contattarlo! Vi ribadisco che questa funzionalità è utile a chi lavora su un sito insieme ad un gruppo di persone oppure se lavorate da soli ma da postazioni diverse, per cui vi può essere utile tenere traccia dei file in fase di modifica. Va da sé che è necessario che tutti i componenti del team lavorino con Dreamweaver ed aggiungerei che è conveniente disabilitare questa funzione se operate autonomamente, perché vi risparmia tempo e spazio, anche se poco.
Passando invece al **"testing server"**, direi che la scelta più importante è quella del tipo di modello server-side che utilizzate: questa opzione dipende unicamente dal vostro provider e dai servizi installati sulle macchina che ospita le vostre pagine. Da questa scelta dipendono a cascata tutte le funzionalità di generazione codice lato server che Dreamweaver mette a disposizione e di cui avremo modo di parlare diffusamente più avanti nella serie. Potete utilizzare un server che supporta ASP (sia con JScript che con VBScript), oppure la nuova piattaforma .NET (con Visual Basic o C#), nonché, per uscire dall'ambito Microsoft, il linguaggio di scripting PHP in appoggio

Fig. 3 - Creazione un nuovo sito senza wizard

ad un database mySQL, o ancora la piattaforma Java Enterprise Edition (con JSP) e quella Macromedia con ColdFusion (nuova versione o in compatibilità con Dreamweaver 4). Nel gruppo di pannelli delle applicazioni (Application, vedi **Figura 4**) troverete una serie di comandi per creare automaticamente script di accesso ai dati e ai loro rendering sulla pagina, oltre ad una serie di server behaviour estensibile (li potete creare voi o scaricare dal sito della Macromedia). Comunque, come dicevo, di questo parleremo poi... Per ora aggiungiamo solo che, anche in questo caso, potete specificare le modalità d'accesso al server, anche se per default vengono riutilizzate quelle impostate per il web server, visto che normalmente si tratterà della stessa macchina.

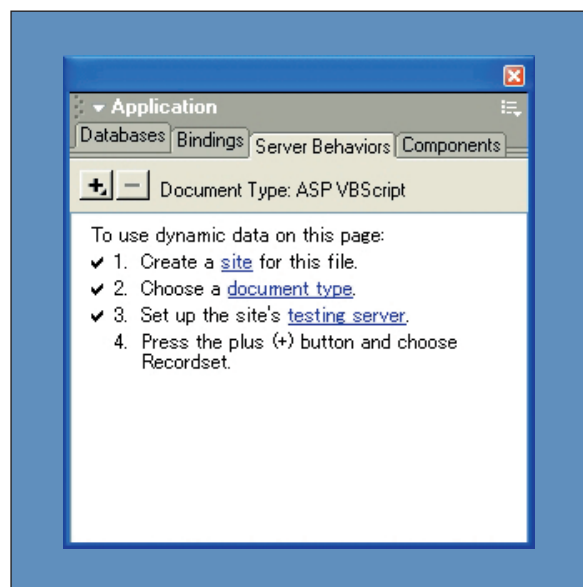



Fig. 4 - Il pannello delle risorse applicative (creazione di codice server-side di accesso ai dati)

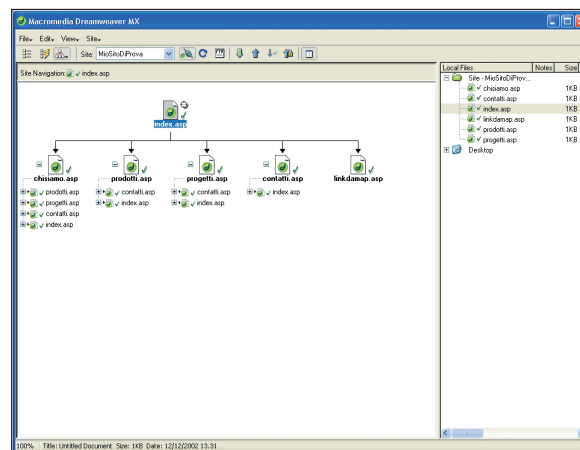
GESTIRE LE RISORSE

Prima di vedere il nostro sito finalmente creato (come progetto, s'intende – magari bastassero pochi click per un sito vero e proprio!), parliamo ancora di un paio di opzioni che si possono impostare.

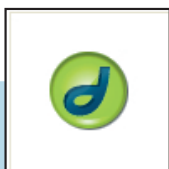
Innanzitutto introduciamo il concetto delle **note di design**: se abilitate questa caratteristica, avrete la possibilità di associare dei commenti personali ai file su cui lavorate. Queste informazioni supplementari sono memorizzate localmente, ma se lavorate in un team potete far sì che le note siano condivise (tramite l'upload su server) con tutti i membri della squadra di lavoro. Tenete solo a mente che anche questa funzione (come il check-out), sebbene molto comoda può rivelarsi un inutile costo di performance per il prodotto quando è attiva senza essere usata. Un'altra pratica capacità di Dreamweaver sui siti è il **cloaking** (da "cloak" che significa mantello): le operazioni di aggiornamento e sincronizzazione dei file tra il server e la copia locale eseguite in blocco (su tutto il sito o su directory, piuttosto che su un singolo file alla volta) possono spesso essere rallentate da grossi file di immagine o di dati o quant'altro che in realtà di aggiornamento non avrebbero bisogno.

Grazie al cloaking è possibile specificare dei gruppi di file (tramite l'estensione) che devono essere esclusi da operazioni batch (cioè, eseguite in blocco), oppure – una volta che l'opzione di **cloaking** è abilitata – si possono indicare specifiche directory che devono anch'esse essere escluse (non è invece possibile richiedere l'esclusione a livello del singolo file). Tutto il materiale filtrato dal **cloaking** potrà ovviamente essere manualmente aggiornato, sincronizzato e scaricato, senza che la sua presenza vada a rallentare le stesse attività quando siano svolte sul sito nel suo complesso.

Infine, un'altra caratteristica importante del prodotto è il **map layout**, che vi consente – a partire da una pagina principale che indicate nella finestra avanzata di gestione del progetto (quella della **Figura 3**, per intenderci) – di visualizzare la struttura del vostro sito evidenziando i legami tra i vari file che lo compongono.

Fig. 5 - La map view del pannello Site

Piuttosto che spiegare cosa intendo dire, comunque, credo sia più semplice indirizzarvi verso la **Figura 5**, che vi mostra la **Map View** del pannello **Site** espanso (il pulsante **Expand/Collapse** vi consente di fare il passaggio tra la vista della **Figura 5** e quella della **Figura 1**: è l'ultima della toolbar del pannello).

Sebbene non si vedano nell'immagine, nella mappa compaiono anche tutti i link a GIF, JPEG, applet e a qualunque altro tipo di risorsa identificabile tramite un URL.

Per visualizzare la map view, selezionate la voce corrispondente nella combo box del pannello **Site**, così come lo vedete nella **Figura 1**.

REPORT, CONTROLLI, E ORGANIZZAZIONE

Lavorando su un sito di grandi dimensioni è facile perdere il senso di quali risorse si abbiano a disposizione e di come siano utilizzate. Per venirci incontro in questo senso, Dreamweaver raccoglie tutto il materiale che è presente nella directory del progetto e lo organizza per tipo di risorsa (immagini, video, link, etc) nel pannello degli **Assets** (che significa, appunto, qualcosa di simile a risorsa, ma con un accento sulla sua importanza e il suo valore).

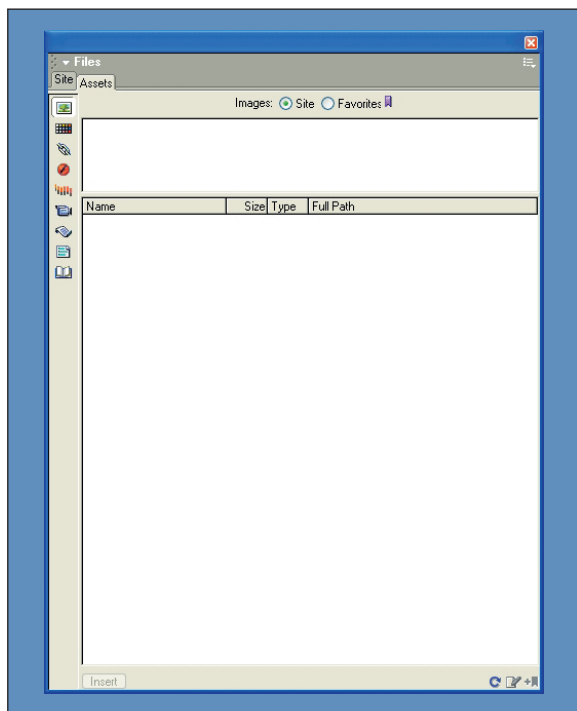



Fig. 6 - Il pannello degli Assets. Qui saranno elencate tutte le risorse utilizzate dal nostro sito.

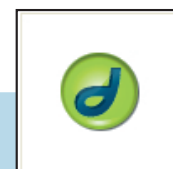
Il pannello lo potete vedere nella **Figura 6**, con tutte le tipologie sulla sinistra, un pannello di preview della risorsa in alto e l'elenco in basso: troverete le immagini, i link (nel senso di URL), i video, i filmati Flash, gli script lato client, e perfino i colori che sono utilizzati nelle pagine HTML del vostro sito. Potete creare una lista di preferiti per ogni categoria, così da avere un accesso più rapido a quelle risorse che usate più di frequente, e con un semplice drag&drop potrete riutilizzare qualunque elemento trascinandolo su una pagina nuova. È da notare che l'ultima tipologia della toolbar verticale (le librerie) vi consente di creare degli elementi di una pagina (blocchi di HTML) che potete poi riutilizzare in vari file HTML diversi, richiedendo che eventuali modifiche apportate alla libreria vengano automaticamente riportate su tutte le pagine che la utilizzano. Infine, vediamo ancora come Dreamweaver ci permette di tenere sotto controllo tutto quello che facciamo. Parleremo da un lato del **Site Reporter** e dall'altro del **Link Checker.** Entrambi producono un output che viene mostrato nel gruppo di pannelli **Results**, di cui abbiamo già parlato nel numero precedente, rispettivamente in **Site Reports** e **Link Checker.** Il primo viene invocato tramite il menu **Site/Reports…** e vi consente di ottenere una serie di informazioni sul vostro sito: i dati raccolti dipendono da quello che avete richiesto nella finestra di dialogo che si apre dal menu, e può includere cose del tipo il designer che detiene file in check-out, le note di design, ed eventuali avvertimenti sull'HTML quali pagine senza titolo (tag TITLE assente) etc. Il secondo strumento, invece, viene invocato o dal menu **Site / Check Links Sitewide** del pannello **Sites**, oppure con il menu contestuale **Check Links** sull'elenco dei file sempre nello stesso pannello. Lo scopo di questa funzione è quello di verificare tutti i link all'interno del sito per assicurarsi che puntino a pagine effettivamente esistenti: eventuali problemi sono categorizzati in **External Links** (non controllati in quanto non verificabili), **Orphaned Links** (pagine esistenti a cui nessuno fa riferimento: di solito sono inutili e da cancellare) e **Broken Links** (link verso pagine del sito che non esistono e che quindi vanno sistemati). Una combo box nel pannello **Link Checker** vi dà la possibilità di filtrare per tipologia di anomalia, mentre un doppio click su un elemento dell'elenco vi apre l'editor della pagina con il problema, pronta per la vostra revisione!

CE N'È ANCORA?

Oggi abbiamo scorso le principali funzioni che ci consentono di organizzare il nostro materiale da lavoro web… Abbiamo parlato di siti e delle risorse che contengono, abbiamo visto come Dreamweaver ce le organizza in assets, vi ho parlato di alcune proprietà dei progetti web e delle possibilità di controllare i link e l'HTML dei vostri file.

Ma Dreamweaver offre molto di più: adesso che avete creato un sito (o progetto, come volete!) potete utilizzare anche i template, una potente feature del prodotto che garantisce riutilizzabilità e facile mantenibilità alle vostre pagine, oltre ai server behaviour di cui si è detto, cioè tutto quel codice server-side generato automaticamente per l'accesso a database. E poi per quello che riguarda le pagine di per sé, ancora dobbiamo parlare di timeline, CSS, ed altre potenti caratteristiche di Dreamweaver.

Insomma, ce n'è ancora abbastanza per potervi chiedere di restare ancorati a queste pagine per qualche altro numero… A presto!

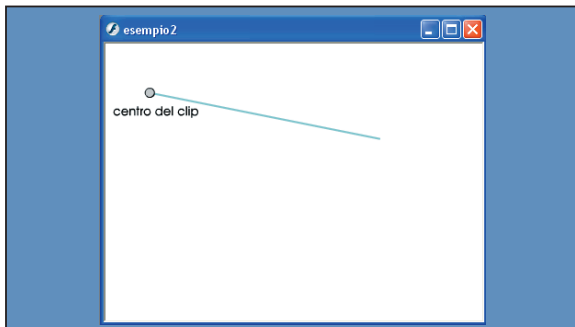
Federico Mestrone

Creare una lavagna interattiva con ActionScript

Fino a qualche tempo fa disegnare figure geometriche usando solo poche righe di codice era una prerogativa di altri linguaggi di programmazione. Oggi, grazie ai nuovi metodi messi a disposizione da ActionScript, è possibile - con estrema facilità - creare linee, curve, forme e riempimenti senza coinvolgere gli strumenti di disegno del programma.

SUL CD

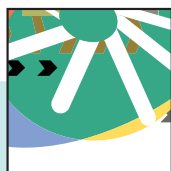
\soft\codice
\drawing.zip

I drawing methods sono metodi attraverso i quali è possibile disegnare curve vettoriali in Flash. I vantaggi che ne conseguono sono fondamentalmente due: in termini di KB, la possibilità di risparmiare risorse del sistema e, in termini visivi, la facoltà di generare effetti interattivi molto interessanti. I metodi relativi al disegno devono sempre essere riferiti ad un clip, con una sintassi del tipo **nomeClip.metodo();** e definiscono, attraverso l'utilizzo di coordinate, la possibilità di disegnare sullo stage. In altre parole, le curve e le figure vengono disegnate all'interno di un clip filmato il cui nome va specificato quando si applica il metodo. Il clip in questione può essere vuoto (cioè privo di qualsiasi elemento grafico) o anche pieno.

Fig. 1- Attraverso i relativi drawing methods abbiamo generato un semplice segmento all'interno dell'istanza del clip filmato clip1.

Elenchiamo nella **Tabella 1** i drawing methods. Facciamo subito un breve esempio per entrare nel vivo (tutti i sorgenti relativi agli esempi si trovano nella cartella **\soft\codice\drawing.zip** del CD-Rom allegato la rivista, mentre è possibile visionare la versione on line del tutorial che illustreremo nel corso di questo articolo, all'indirizzo **www.dynamicdesign.it/drawing**). Apriamo Flash MX e, per creare un clip, seguiamo il percorso **Inserisci>Nuovo** simbolo, selezioniamo l'opzione **clip**, diamogli come nome **clip1** e, infine, premiamo **Ok**. Apriamo la libreria **(Finestra>Libreria)**, trasciniamo il nostro clip sullo stage e, tramite la barra delle proprietà, scegliamo **clip1** come nome istanza. A questo punto, inseriamo nel primo fotogramma le seguenti tre righe di codice:

```
1 _root.clip1.lineStyle(2,0x22899,50);
2 _root.clip1.moveTo(0,0);
3 _root.clip1.lineTo(250,50);
```

Apriamo il filmato Flash **esempio1** e vediamone un'anteprima attraverso il menu **Controlli>Prova filmato** (vedi **esempio1.fla** nella cartella del CD-Rom). In pratica abbiamo creato una linea senza utilizzare gli strumenti di disegno: il primo metodo specifica che la nostra linea avrà come spessore 2 (il valore può andare da 0 a 255), come colore RGB un valore esadecimale #228899 e, come trasparenza, un valore alfa pari a 50. È interessante notare che il colore viene indicato con lo stesso procedimento usato per i metodi dell'oggetto **Color** di **ActionScript**, ovvero con **"0x"** al posto del simbolo "#", che si usa in genere nell'HTML. Come è facile intuire osservando il breve codice, la seconda riga indica le coordinate del punto di partenza del nostro segmento, mentre la terza fissa le coordinate del secondo punto. Le coordinate fanno riferimento al sistema di assi che ha come vertice il centro del clip, per cui il primo punto, che ha come valore **x** ed **y** pari a 0, sarà posizionato esattamente al centro del clip filmato **clip1**. Se al posto del clip vuoto ne usiamo uno già composto da un elemento grafico (ad esempio un piccolo cerchio posto al centro della relativa linea temporale) il risultato non cambia più di tanto: semplicemente, l'elemento grafico creato con **ActionScript** si aggiunge all'elemento grafico preesistente. Se invece volessimo relazionare la posizione dei disegni al centro dello stage, basterebbe riferire i metodi direttamente alla **_root**, con una sintassi del tipo **_root.nomeMetodo();** (vedi **esempio3.fla** nella cartella del CD-Rom). Tuttavia, poiché il più delle volte è utile disegnare figure da zero in clip privi di elementi grafici, può essere comodo impiegare il metodo **createEmptyMovieClip()**, che consente la creazione di un clip vuoto attraverso **ActionScript**. Ad esempio, nel caso volessimo sperimentare il metodo **curveTo()** per la creazione di una curva di Bezier (vedi **esempio4.fla**), potremmo fare a meno di creare manualmente il nostro clip vuoto, inserendo nel primo fotogramma un listato con una sintassi del tipo:

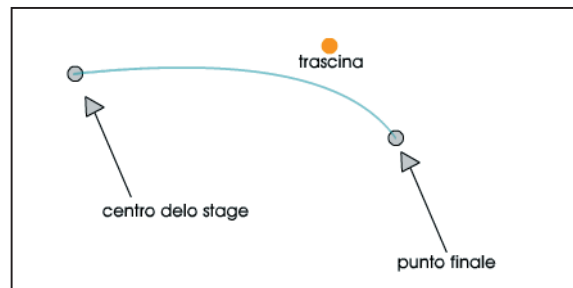
```
1 _root.createEmptyMovieClip("clip1",1)
2 _root.clip1.lineStyle(2,0x228899,50);
3 _root.clip1.moveTo(0,0);
4 _root.clip1.curveTo(60,45,250,50);
```

In questo modo viene creato un clip sul livello specificato, analogamente a quanto accade utilizzando i metodi **attachMovie()** o **duplicateMovieClip()**. Poiché non abbiamo indicato le coordinate del clip vuoto, questo, di default, verrà posizionato in corrispondenza dell'origine dello stage, cioè nel punto coincidente con l'angolo in alto a sinistra dell'area di lavoro.

Con alcuni piccoli ritocchi al codice appena illustrato, possiamo esplorare le potenzialità interattive dei drawing methods (vedi **esempio5.fla**): ritoccando il listato e aggiungendo un clip filmato denominato maniglia, è possibile modificare interattivamente la curva di Bezier.

```
1 _root.onEnterFrame=function(){
2 _root.createEmptyMovieClip("clip1",1)
3 _root.clip1.lineStyle(2,0x228899,50);
4 _root.clip1.moveTo(0,0);
5 a=_root.maniglia._x;
6 b=_root.maniglia._y;
7 _root.clip1.curveTo(a,b,250,50);
8 }
9 _root.maniglia.onPress=function(){
10 _root.maniglia.startDrag(true);
11 }
12 _root.maniglia.onRelease=function(){
13 _root.maniglia.stopDrag();
14 }
```

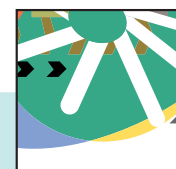
nome	utilizzo
lineStyle();	Indica le caratteristiche della linea e prevede tre parametri: spessore, colore, trasparenza.
moveTo();	Indica le coordinate del punto iniziale della linea e prevede due parametri (x1 e y2).
lineTo();	Indica le coordinate del punto finale della linea e prevede due parametri (x2 e y2).
curveTo();	Specifica che la linea sarà una curva di Bezier; in genere sostituisce lineTo() e prevede quattro parametri (xc,yc,x2,y2). x2 e y2 indicano le coordinate del punto finale, xc e yc, invece, le coordinate del "nodo" a cui fa riferimento la curva di Bezier.
beginFill();	Indica il colore di riempimento e la trasparenza di una porzione specificata, e prevede due parametri (valoreRGB, valoreAlpha).
beginGradientFill();	Indica un colore di riempimento sfumato, che può esser radiale o lineare. Questo metodo prevede quattro parametri: tipo di riempimento, array dei colori usati, array che indica il valoreAlpha di ogni colore, array che distribuisce i vari colori, matrice che definisce le caratteristiche del riempimento sfumato.
endFill();	Indica dove finisce la porzione da colorare e lavora in associazione di beginFill() o beginGradientFill(); non prevede parametri.
clear();	Consente di eliminare quanto disegnato nel clip specificato; non prevede parametri.

Tab. 1 - Elenco dei drawing methods disponibili in Flash MX.

Fig. 2 - Trascinando il cerchio al centro, vengono cambiate interattivamente le coordinate del punto di controllo che regola l'inclinazione della curva di Bezier.

La creazione della curva è stata associata all'evento **onEnterFrame**, il quale farà in modo che i codici presenti tra la riga 2 e la riga 8 siano eseguiti ad ogni lettura del fotogramma (e quindi continuamente). Inoltre, vengono utilizzati gli eventi **onPress** e **onRelease**, entrambi associati al clip maniglia affinché, ad ogni pressione e rilascio, sul clip siano attivati i metodi **startDrag()** e **stopDrag()**, che gestiscono rispettivamente il trascinamento e la fine del trascinamento del clip specificato. Le coordinate del clip trascinabile costituiscono i parametri che regolano la curva di Bezier (righe 5 e 6), per cui, quando l'utente sposta il clip maniglia, cambiano le caratteristiche della curva. Questo meccanismo di interazione è alla base della lavagna interattiva che creeremo tra breve. Tuttavia, prima di illustrare il tutorial vero e proprio, è opportuno spendere due parole sui metodi che consentono di creare porzioni dotate di riempimenti. Con i drawing methods, infatti, è possibile generare non solo semplici linee, ma anche figure più complesse. Per fare un esempio, analizziamo il seguente codice, presente nel primo fotogramma del filmato **esempio6.fla:**

```
1 _root.createEmptyMovieClip("clip1",1)
2 _root.clip1.lineStyle(2,0xFF0000,50);
3 //_root.clip1.beginFill(0xFF6600,100);
4 _root.clip1.moveTo(120,110);
5 _root.clip1.lineTo(210,185);
6 _root.clip1.lineTo(85,175);
7 _root.clip1.lineTo(120,110);
8 //_root.clip1.endFill();
```

Semplicemente, è stato creato un triangolo tracciando tre linee e facendo in modo che la terza terminasse esattamente dove viene generata la linea di partenza. Le righe 3 e 8 del listato sono precedute dal doppio slash // che - analogamente a quanto avviene in altri linguaggi di programmazione - ne impedisce l'esecuzione, trasformando il codice che segue in un commento. Rendendo attive le righe in questione, cioè cancellando le doppie barre iniziali, si ottiene una figura piena, poiché viene definita una porzione che inizia con **beginFill()** - il metodo che specifica un colore e una trasparenza - e termina con **endFill()**. Il discorso si complica quando si vogliono dare dei riempimenti sfumati. Infatti il metodo **beginGradientFill()** non ha un utilizzo altrettanto immediato, a causa dei numerosi parametri previsti, alcuni dei qua-

li abbastanza macchinosi da definire. Analizziamo il listato tratto dal primo fotogramma del filmato **esempio7.fla**:

```
1 tipo="linear";
2 colori=[0xFFFF99,0xFF0000];
3 trasparenza=[100,100];
4 distribuzione=[0,255];
5 matrice={matrixType:"box",x:110,y:120,w:70,h:70,r:45};
6 _root.createEmptyMovieClip("clip1",1);
7 _root.clip1.lineStyle(0x000000,50);
8 _root.clip1.beginGradientFill(tipo,colori,trasparenza,
                                    distribuzione,matrice);
9 _root.clip1.moveTo(120,110);
10 _root.clip1.lineTo(210,185);
11 _root.clip1.lineTo(85,175);
12 _root.clip1.lineTo(120,110);
13 _root.clip1.endFill();
```

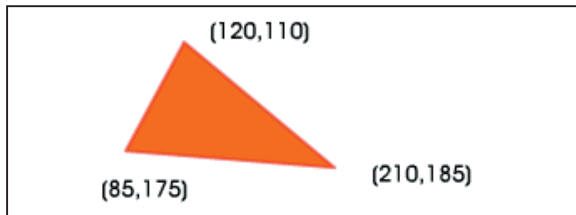



Fig. 3 - Nel momento in cui si usano beginFill() ed endFill(), il programma identifica automaticamente la porzione da riempire a prescindere dalla forma disegnata.

Questo codice permette di attribuire al triangolo realizzato precedentemente un riempimento lineare, che va da un colore giallo chiaro al rosso. Molti dei parametri impiegati sono degli array e, proprio a causa della loro complessità, è necessario definirli separatamente, creando quattro variabili che saranno poi successivamente utilizzate dal metodo **beginGradientFill()**; Il primo parametro che abbiamo passato alla variabile **tipo** prevede solo due valori, che sono rispettivamente **"radial"** o **"linear"**. Il secondo invece è caratterizzato da un array di n elementi, ognuno dei quali costituisce un valore esadecimale - in altre parole indica da quanti e quali colori sarà costituito il nostro riempimento sfumato. Segue poi un array costituito da tanti valori alfa quanti sono i colori utilizzati (nel nostro caso abbiamo dato la massima opacità ad entrambi i colori). Abbiamo poi la miscelazione dei colori all'interno della porzione indicata: impostando questi valori si compie un'operazione analoga al trascinamento dei simboli corrispondenti ai colori di un riempimento sfumato, quando si utilizza il pannello **Mixer** per disegnare in Flash MX. Infine, abbiamo un array associativo che definisce sei elementi. Il primo parametro è

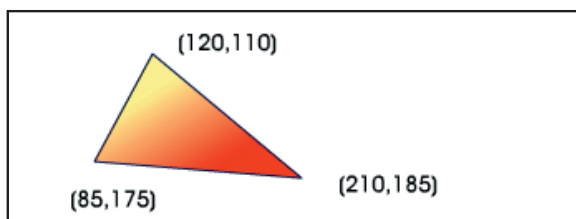
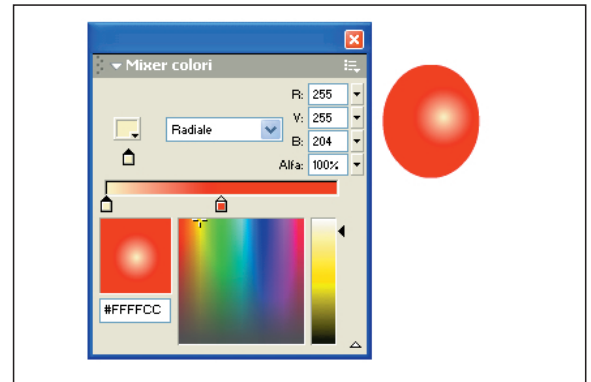
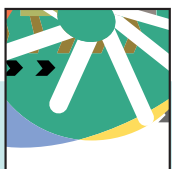



Fig. 4 - Allo stesso triangolo creato in precedenza abbiamo attribuito un riempimento sfumato lineare.

Fig. 5 - Quando si usa il pannello Mixer per creare un riempimento radiale o lineare, è possibile distribuire interattivamente i vari colori.

sempre **box**; il secondo e il terzo definiscono le coordinate dell'angolo superiore sinistro del riempimento sfumato rispetto al centro del clip; Il terzo e il quarto, la larghezza e l'altezza del riempimento. Il quinto parametro indica l'angolo di rotazione del riempimento sfumato (ha senso attribuirgli un valore diverso da zero solo quando si usa un tipo di riempimento lineare). Ultimata la nostra panoramica sui drawing methods, possiamo dedicarci ad un progetto più impegnativo che ci permetta di sfruttare pienamente le possibilità offerte. Nella cartella del cd-rom allegato alla rivista è possibile trovare sia il file completo, denominato **lavagna.fla**, sia una versione "essenziale", denominata **interazione_semplice.fla**. Quest'ultima mostra il principio di funzionamento alla base della lavagna interattiva: infatti, se testiamo il filmato con **Controlli>Prova filmato** noteremo che, cliccando sull'area di lavoro e trascinando con il puntatore del mouse, è possibile disegnare una linea a mano libera. Apriamo il file **interazione_semplice.fla**: ci sono tre livelli (**azioni**, **clip** e **stage**). Nel livello stage è posizionato un pulsante invisibile che copre tutta l'aera di lavoro e ha come nome istanza "stage", mentre, nel livello **clip**, è presente un clip filmato denominato **clip1**. Se apriamo la libreria ed esploriamo **clip1**, notiamo che è composto da due clip, **m1** ed **m2**, costituiti da piccoli cerchi trasparenti, posizionati uno sopra l'altro. Osserviamo il codice presente nel primo fotogramma del livello **azioni**:

```
1 _root.stage.useHandCursor = false;
2 _root.stage.onPress = function() {
3 _root.clip1.lineStyle(5, 0x000000);
4 _root.clip1.moveTo(_root.clip1.m1._x, _root.clip1.m1._y);
5 _root.a = _root._xmouse;
6 _root.b = _root._ymouse;
7 startDrag("_root.clip1.m2");
8 }
9 _root.onEnterFrame = function() {
10 _root.clip1._x = _root.a;
11 _root.clip1._y = _root.b;
12 _root.clip1.lineTo(_root.clip1.m2._x, _root.clip1.m2._y);
13 }
14 _root. stage.onRelease = _root.
                        stage.onReleaseOutside=function () {
15 stopDrag();
16 _root.clip1.clear();
```

```
17 _root.clip1.m2._x = _root.clip1.m1._x;
18 _root.clip1.m2._y = _root.clip1.m1._y;
19 }
```

Nella riga 1, settando la proprietà **useHandCursor** sul valore booleano **false**, viene disabilitata la mano che caratterizza il puntatore quando si passa sul pulsante **stage**. Attraverso l'evento **onPress** (dalla riga 2 in poi), ad ogni pressione del mouse sul pulsante invisibile **stage**, con i metodi **lineStyle** e **moveTo()** viene creata una linea all'interno del clip filmato **clip1**; viene anche specificato che il punto iniziale della linea è contraddistinto dalle coordinate del clip **m1**. Successivamente, sempre in corrispondenza della pressione effettuata dall'utente sul pulsante **stage**, vengono passate le coordinate **x** ed **y** del puntatore a due variabili **a** e **b**. Infine, viene attivato il trascinamento del clip **m2**. Non è stato definito il metodo **lineTo()**, che stabilisce il punto finale della linea, poiché la sua definizione viene effettuata all'interno del blocco di codice gestito dall'evento **onEnterFrame**, a partire dalla riga 9. Qui il metodo **lineTo()** utilizza come coordinate la **x** e la **y** del clip **m2**, che viene trascinato dall'utente, e che quindi cambia continuamente fino a quando l'utente non interrompe il trascinamento: in altri termini, viene generata una piccola linea ad ogni minimo spostamento effettuato. Non solo: l'evento **onEnterFrame** stabilisce le coordinate del **clip1**, attribuendogli i valori delle variabili **a** e **b** usate in precedenza: ad ogni click dell'utente, il clip filmato **clip1** si posizionerà nel punto in cui si trova il mouse (l'origine della linea creata con il trascinamento). Grazie agli eventi **onRelease** e **onReleaseOutside**, quando si rilascia il mouse sul pulsante **stage** o al di fuori della sua area attiva, si interrompe il trascinamento del clip **m2**, che viene riposizionato in corrispondenza del clip **m1** (cioè al centro del clip filmato **clip1**). Invece, con l'azione **clear ()**, viene cancellata la linea creata. Alla fine viene quindi ricreata la condizione iniziale che consente la creazione di una nuova linea. Tuttavia, il meccanismo appena descritto è ben lontano dalla possibilità di creare una vera e propria lavagna interattiva, perché non presenta opzioni di regolazione (cioè non è possibile cancellare o ritoccare quanto disegnato) e soprattutto perché, per evitare di creare una poligono chiuso, la linea viene cancellata ad ogni rilascio del mouse - mentre invece ci sarebbe la necessità di creare una diversa linea che rimanga visibile ad ogni intervento dell'utente. Se vediamo il file **lavagna.swf** all'opera, notiamo che è possibile non solo memorizzare le linee create dall'utente, ma anche intervenire su altri fattori, come ad esempio il colore delle linee. Apriamo **lavagna.fla** nella cartella del CD-Rom e osserviamo la disposizione dei livelli: è identica a quella utilizzata nella file **interazione_semplice.fla**. Infatti, la differenza è tutta nel codice. La tecnica, utilizzata per creare tante linee quanti disegni tracciati dall'utente, è basata sull'uso del metodo **duplicateMovieClip()**. Cioè, ogni volta che l'utente traccia una nuova linea, viene creato un duplicato chiamato **clip1**, distrutto poi con il metodo **removeMovieClip()**, non appena l'utente rilascia il pulsante del mouse: questo consente di non settare i clip **m1** ed **m2** alla fine della creazione di ogni linea perché, ogni volta che si duplica il clip originale posto nel livello clip, la copia tiene conto delle coordinate iniziali dei clip interni. Il passaggio più importante è costituito dalla duplicazione effettuata a partire dalla riga 31: viene creata una copia della linea appena generata, ogni volta che l'utente rilascia il tasto del mouse. Questo meccanismo tiene conto anche delle disposizione progressiva delle varie linee sui relativi livelli. A regolare l'incremento dei vari livelli ci pensa la variabile **n**, settata inizialmente su 1 e poi incrementata in relazione ad ogni nuova linea creata. Sia il clip **maschera**, che funge da cornice della lavagna, sia il clip **matita**, sono stati posizionati - grazie al metodo **attachMovie()** - su un livello che prevede un numero altissimo di linee disegnate, allo scopo di evitare che le linee tracciate dall'utente coprano il bordo della lavagna. Per creare un tasto che funga da gomma, è sufficiente settare l'evento **onRelease**, riferito al pulsante gomma, con una sintassi del tipo:

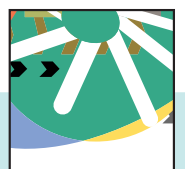
Fig. 6 - utilizzando lo stesso meccanismo di programmazione visto in precedenza, si può creare una vera e propria lavagna interattiva.

```
1 _root.maschera.gomma.onRelease=function(){
2 colore=0xFFFFFF;
3 spessore=8;
4 orientamento=180;
5 }
```

```
20 _root.stage.onPress=function(){
21     _root.n++;
22 duplicateMovieClip("_root.clip", "clip1", 1+n);
23     _root.clip1.lineStyle(spessore, colore);
24     _root.clip1.moveTo(_root.clip1.m1._x, _root.clip1.m1._y);
25     _root.a = _root._xmouse;
26     _root.b = _root._ymouse;
27     startDrag("_root.clip1.m2");
28 }
29 _root.stage.onRelease=_root.stage.onReleaseOutside= function(){
30 stopDrag();
31     duplicateMovieClip("_root.clip1", "traccia"+_root.n, _root.n);
32     removeMovieClip(_root.clip1);
33 }
```

Fig. 7 - Con il metodo *duplicate MovieClip()* viene creato un clip, il cui nome è composto dalla parola "traccia" seguita da un numero progressivo n.

dove le variabili colore e spessore sono i parametri passati al metodo **lineStyle()**, che definisce le caratteristiche della linea ad ogni pressione del mouse, mentre la variabile orientamento regola la proprietà **_rotation** del clip matita (che si capovolge in modalità gomma).

Maurizio Battista

Web Poll con ASP.NET & Web Matrix

Realizziamo un'applicazione web per effettuare sondaggi grazie alla semplicità e alla potenza del binomio ASP.NET e Web Matrix.

SUL CD
\soft\codice
\Web_Poll.zip

L'interazione tra l'autore e il visitatore di un sito web è un fattore che può determinarne il successo. Nel panorama innovativo delle trovate per l'intrattenimento degli utenti si sta affacciando da alcuni tempi la "moda" dei sondaggi web (web poll). La possibilità di dare un giudizio del tutto anonimo su tematiche interessanti per i visitatori e la curiosità di confrontare con uno sforzo, che si limita ad un click, la propria idea con quella degli altri utenti, sono fattori di stimolo psicologico che invitano all'uso di quest'utile strumento di indagine. Il sondaggio inoltre può avere domande specifiche atte al miglioramento del sito stesso, costituendo quindi un possibile metodo di **feedback**. Le possibili implementazioni dell'applicazione Web sono molteplici, ma in quest'articolo punteremo alla semplicità derivante dall'uso di Web Matrix come ambiente di sviluppo ASP.NET, e la sua totale integrazione con **MSDE** (Microsoft Sql Server 2000 Desktop Engine). L'applicazione web è costituita da tre pagine: **poll.aspx**, **vota.aspx** e **risultati.aspx**. L'utente potrà scegliere dalla pagina **poll.aspx** uno dei sondaggi "attivi" (non ancora scaduti) oppure visualizzare i risultati dei sondaggi conclusi (la cui data di "chiusura" delle votazioni è passata). **Vota.aspx** permette la visualizzazione delle varie possibilità di risposta alle domande dei sondaggi, e presenta un pulsante per esprimere il proprio voto. Per evitare voti multipli in un solo giorno, da parte di uno stesso utente, dovrà effettuare anche controlli sull'indirizzo IP. Subito dopo aver votato, l'utente viene indirizzato verso la corrispondente pagina **risultati.aspx** per osservare i voti degli altri utenti in risposta al sondaggio appena votato.

Fig. 1 - Le funzionalità di Data Administration richiedono componenti client di SQL presenti in MSDE o SQL Server. Viene fornito un link al download (29,5 MB) di MSDE

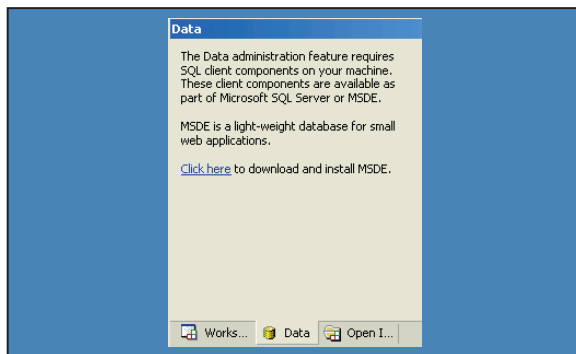



Fig. 2 - I primi passi per la creazione del database Web Poll

IL DATABASE

Web Matrix ha al suo interno un utilissimo database manager che interagisce con MSDE o eventualmente SQL Server 2000, e permette la creazione di un database da zero. Il tab Data situato a destra dell'ambiente di sviluppo, ci invita al download gratuito di MSDE nel caso in cui non sia ancora installato. Subito dopo l'installazione e la configurazione di MSDE (avvenuta seguendo le istruzioni fornite nella pagina web del download) all'interno di Web Matrix siamo pronti alla creazione del database che conterrà le informazioni del web poll.
Dobbiamo creare due tabelle, **Sondaggi** e **Votanti**; quest'ultima verrà

Tabella Sondaggi

CAMPO	TIPO
ID_SOND	Int, Richiesto, Chiave Primaria e Autoincremento
Domanda	Text, Richiesto
DataInizio	DateTime, Richiesto
DataFine	DateTime, Richiesto
Opzione1, Opzione2, Opzione3, Opzione4, Opzione5, Opzione6	Text
Voti1, Voti2, Voti3, Voti4, Voti5, Voti6	Int

Tabella Votanti

CAMPO	TIPO
ID	BigInt, Richiesto, Chiave Primaria e Autoincremento
ID_SOND	Int, Richiesto
IP	Text, Richiesto
Data	DateTime, Richiesto

Fig. 3 - La creazione del tutto intuitiva della tabella Sondaggi

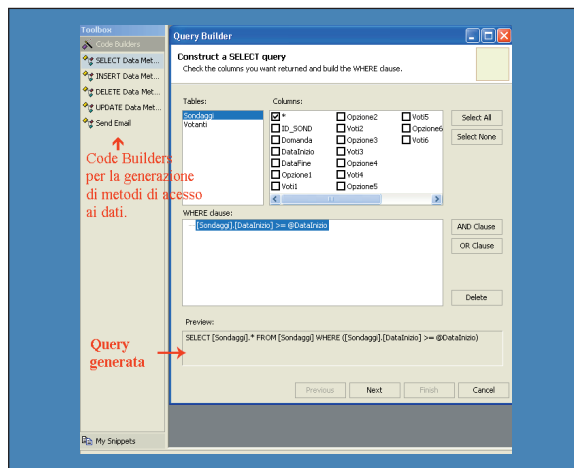
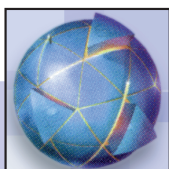
utilizzata per evitare voti multipli da parte di uno stesso indirizzo IP. Facendo click sul tab **data**, e successivamente sul pulsante di **"new connection"**, apparirà una finestra di dialogo per la connessione a un database esistente o la creazione di uno nuovo. Un click su **"Create new database"** all'interno di quella finestra, provocherà l'apertura di un altro form richiedente il nome del database e nel quale dobbiamo inserire **WebPoll**. A questo punto il tab data mostrerà un menù gerarchico contenente una connessione al database, e due voci **"Tables"** e **"Stored Procedures"**. Non faremo uso di **Stored Procedure**, per cui selezionando **Tables** e facendo click sul pulsante **New Item** (accanto a **new connection**) si aprirà una finestra di dialogo per la creazione di una tabella. L'inserimento delle informazioni sulla tabella e i campi è sorprendentemente intuitivo, tanto da non meritare ulteriori spiegazioni.
Dobbiamo ripetere la procedura di creazione anche per la tabella **Votanti**. I campi che bisogna inserire e le loro proprietà sono mostrati nelle **Tabelle 1** e **2**. Ogni sondaggio è identificato da un suo id nella tabella **Sondaggi (ID_SOND)**, **Opzione1** sino a **Opzione6**, sono sei campi necessari per contenere le risposte al testo del sondaggio contenuto in **Domanda**. I campi che vanno da **Voti1** sino a **Voti6**, contengono il numero di voti corrispondenti alle rispettive opzioni. In fine, **DataInizio** e **DataFine** indicano l'intervallo temporale entro il quale il sondaggio risulta "attivo" e non ancora concluso. La tabella **Votanti**, invece, ha un ID per ogni voto, **ID_SOND** che esprime il sondaggio a cui si sta rispondendo, **IP** è l'IP dell'utente che ha votato e **Data** è il campo contenente la data del voto. In questo modo, potremo controllare se un utente (identificato dal suo IP) ha già votato oggi (**Data**) per un determinato sondaggio (**ID_SOND**).
Ogni sondaggio potrà quindi avere sei risposte possibili e questo può essere considerato un limite perché si potevano stabilire delle relazioni tra tabelle e creare una struttura più flessibile con un numero dinamico di opzioni. Il database **WebPoll**, pur essendo lineare e volutamente semplice, risulta comunque una buona soluzione per la maggior parte delle esigenze dei sondaggi. Inoltre la sua immediatezza permette di focalizzare l'attenzione sul codice ASP.NET e gli strumenti offerti da Web Matrix.

ARCHIVIO SONDAGGI

La pagina **poll.aspx** si comporta come archivio sondaggi. Dovrà visualizzare e separare i sondaggi che sono ancora attivi e quindi "votabili", da quelli la cui **DataFine** è già passata. Per i primi dovrà fornire un link alla pagina **vota.aspx**, passando come parametro l'ID del sondaggio su cui si è fatto click, mentre per i secondi dovrà semplicemente mostrare la pagina dei risultati parametrizzata sempre con l'ID. In **Figura 4** è mostrato come appare la pagina una volta completata, in presenza di alcuni sondaggi nel database.

Fig. 4 - L'archivio dei sondaggi attivi e conclusi

Nella stesura del codice avremo bisogno di generare due funzioni che effettuino delle query sulla tabella **Sondaggi**, determinando i sondaggi attivi (**MyQueryAttivi**) e quelli conclusi (**MyQueryConclusi**). Entrambe le funzioni accettano come parametro la data odierna e restituiscono un oggetto **DataReader**. Creata la pagina **poll.aspx** in Web Matrix, si faccia click sulla veduta **"Code"** della finestra principale. L'ambiente di sviluppo ci aiuta nel compito di creare le due funzioni mediante degli speciali wizard (i **"Code Builders"**) presenti nella toolbox a sinistra dell'ambiente. Facendo drag&drop di **"Select Data Method"** nella pagina, apparirà un comodo wizard per la creazione intuitiva e visuale della query.

Fig. 5 - Il wizard Select Data Method per la generazione di una delle due funzioni che effettuano la query di select sui sondaggi, per determinare quelli attivi o conclusi

In questo caso il codice generato è già pronto e non bisogna effettuare alcuna modifica manuale. A meno di un operatore di confronto, le due funzioni sono pressoché identiche, per cui bisognerà ripetere gli inserimenti nel wizard, sia per **MyQueryAttivi** sia per **MyQueryConclusi**. Il codice sorgente della funzione **MyQueryAttivi** è:

```
Function MyQueryAttivi(ByVal dataFine As Date)
                    As System.Data.SqlClient.SqlDataReader
  Dim connectionString As String = "server='localhost';
                trusted_connection=true; Database='WebPoll'"
  Dim sqlConnection As System.Data.SqlClient.SqlConnection
   = New System.Data.SqlClient.SqlConnection(connectionString)
  Dim queryString As String = "SELECT [Sondaggi].* FROM
                   [Sondaggi] WHERE ([Sondaggi].[DataFine] >
                     = @DataFine) ORDER BY DataFine DESC"
  Dim sqlCommand As System.Data.SqlClient.SqlCommand = New
     System.Data.SqlClient.SqlCommand(queryString, sqlConnection)
  sqlCommand.Parameters.Add("@DataFine",
              System.Data.SqlDbType.DateTime).Value = dataFine
  sqlConnection.Open
  Dim dataReader As System.Data.SqlClient.SqlDataReader =
                              sqlCommand.ExecuteReader(
               System.Data.CommandBehavior.CloseConnection)
  Return dataReader
 End Function
```

Il parametro **@DataFine** della query **SELECT**, viene associato a quello passato dalla funzione mediante l'istruzione **sqlCommand.Parameters.Add"**. Infine la funzione ritorna il DataReader contenete i record risultanti dalla query. La condizione **[Sondaggi].[DataFine]** >= **@DataFine** indica che la data del sondaggio deve essere maggiore di quella odierna (per cui non ancora scaduta). **MyQueryConclusi** ha la condizione opposta (**<**). Il segno uguale può essere posto a piacimento, in base alla propria volontà di includere o meno il giorno stesso di scadenza. Al caricamento della pagina verrà eseguita la routine di gestione dell'evento:

```
Sub Page_Load(sender as Object, e as EventArgs)
Dim drConclusi as System.Data.SqlClient.SqlDataReader =
                         MyQueryConclusi(DateTime.Today())
Dim drAttivi as System.Data.SqlClient.SqlDataReader =
                           MyQueryAttivi(DateTime.Today())

Response.Write("<h1 align=""center""> Web Poll</h1>")
Response.Write("<p class=""titoletto"">Sondaggi attivi
        <hr color=""#336699"" align=""left"" width=""25%"">")
while (drAttivi.read())
   Response.Write(drAttivi("DataFine") & " - ")
   Response.Write("<a href=""vota.aspx?id="& drAttivi(
         "ID_SOND")  & """>" & drAttivi("Domanda") & "</a>")
   Response.Write("<br>")
end While
Response.Write("</p>")
Response.Write("<br>")

Response.Write("<p class=""titoletto"">Sondaggi conclusi
        <hr color=""#336699"" align=""left"" width=""25%"">")
while (drConclusi.read())
   Response.Write(drConclusi("DataFine") & " - ")
   Response.Write("<a href=""risultati.aspx?id="& drConclusi(
         "ID_SOND") & """>" & drConclusi("Domanda") & "</a>")
   Response.Write("<br>")
end While
Response.Write("</p>")
End Sub
```

DateTime.Today() indica la data odierna fissata all'ora **00.00.00** e viene passata alle due funzioni il cui risultato viene assegnato a due oggetti DataReader: **drAttivi** e **drConclusi**. Mediante il ciclo while si recuperano tutti i record contenuti nel DataReader e vengono visualizzati esclusivamente i campi **DataFine**, e **Domanda**. Per la generazione del link col tag html **<a>** viene recuperato anche l'**ID_SOND**. Come facilmente osservabile, i sondaggi conclusi, a differenza di quelli attivi, non hanno come "target" del link la pagina di votazione parametrizzata mediante l'id del sondaggio in questione, ma la pagina dei risultati dello stesso. Nelle query effettuate dalle funzioni abbiamo selezionato tutti i campi della tabella **Sondaggi**, mediante il carattere di asterisco (*****) per permettere la possibile visualizzazione anche di altri campi all'interno della pagina **poll.aspx**; tuttavia è chiaro che per la nostra implementazione sono sufficienti i tre campi **ID_SOND**, **Domanda** e **DataFine.**

LA PAGINA VOTA.ASPX

La pagina **poll.aspx**, nel caso dei sondaggi attivi, rimanda alla pagina di votazione **vota.aspx**, alla quale "passa" l'ID del sondaggio come parametro visibile e recuperabile mediante il metodo **QueryString.** L'indirizzo presente nella barra degli indirizzi a cui verremo re-indirizzati avrà un formato simile al seguente: **http://localhost:8080/vota.aspx?id=2** . Il valore 2 del parametro id, indica che stiamo accedendo alla votazione del sondaggio avente **ID_SOND = 2** nella tabella **Sondaggi**. La pagina dovrà apparire in maniera analoga a quella mostrata in figura (salvo i dati fittizi inseriti).

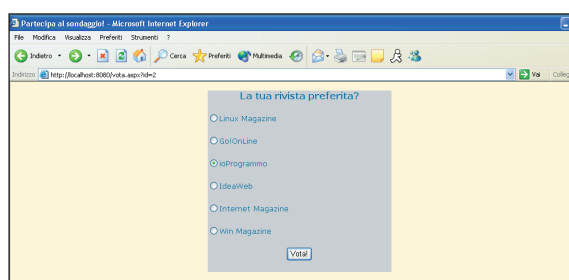
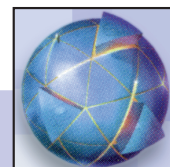



Fig. 6 - La pagina per esprimere il proprio voto ad un sondaggio

Facendo click sulla veduta **Design** della finestra principale dell'IDE, appare nella toolbox di sinistra il gruppo **Web Controls**. Dovremo effettuare dei **Drag&Drop** di un pannello, una label, sei **radioButton** e di un button. I nomi da assegnare sono rispettivamente, **pnlVotazione** (facoltativo), **lblDomanda**, **rb1, rb2, rb3, rb4, rb5, rb6 e btnVota.** Potremo sfruttare appieno il gruppo proprietà di Web Matrix senza agire sul codice, per modificare le principali caratteristiche di questi controlli. In particolare è necessario che alla proprietà **GroupName** dei radiobutton sia assegnato il valore **"Opzioni"**. Per quanto riguarda il pulsante, dovremo modificare manualmente il codice (**tasto destro -> Edit Tag**) ottenendo:

```
<asp:Button id="btnVota" onclick="btnVota_Click" runat=
                         "server" Text="Vota!"></asp:Button>
```

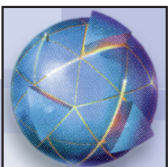
In particolar modo **onclick="btnVota_Click"**, permette di attivare la routine di gestione dell'evento denominata **btnVota_Click** che analizzeremo nel seguito. Al caricamento della pagina, dovremo in qualche modo "associare" la domanda e le opzioni del sondaggio indicato dal parametro **ID**:

```
Sub Page_Load(sender as Object, e as EventArgs)
  Dim dr As System.Data.SqlClient.SqlDataReader
  Dim id As Integer = Int32.Parse(Request.QueryString("id"))
  if (Not Page.IsPostback) then
    'datareader risultante dal metodo MyQuerySondaggi
    'avente id del sondaggio come parametro del metodo e della
    'query sql
    dr = MyQuerySondaggi(id)
    while (dr.Read())
      lblDomanda.Text = dr("Domanda")
      rb1.Text = dr("Opzione1")
      rb2.Text = dr("Opzione2")
      rb3.Text = dr("Opzione3")
      rb4.Text = dr("Opzione4")
      rb5.Text = dr("Opzione5")
      rb6.Text = dr("Opzione6")
    end while
  end if
End Sub
```

ID identifica il sondaggio e viene recuperato con il metodo **QueryString**. **MyQuerySondaggi** è del tutto analoga alle precedenti funzioni viste per la pagina **poll.aspx** e creata mediante i wizard di Web Matrix; tuttavia il parametro della query e della funzione è l'ID del sondaggio. Al solito viene restituito un DataReader che ci permette di ottenere il valore dei vari campi richiesti e di assegnarli alle proprietà **Text** dei Web Controls. La parte più complessa risiede nella gestione dell'evento click:

```
Sub btnVota_Click(sender As Object, e As EventArgs)
   Dim id As Integer = Int32.Parse(Request.QueryString("id"))
   Dim v1,v2,v3,v4,v5,v6, updated, inserted As Integer
   Dim dr As System.Data.SqlClient.SqlDataReader =
                                        MyQuerySondaggi(id)
   Dim opz As String = Request.Form("Opzioni")
   Dim ip As String = Request.ServerVariables("REMOTE_ADDR")
   Dim oggi As DateTime = DateTime.Today()
   if VotoValido(id) then
      'recupero voti attuali
      while (dr.Read())
         v1 = dr("Voti1")
         v2 = dr("Voti2")
         v3 = dr("Voti3")
         v4 = dr("Voti4")
         v5 = dr("Voti5")
         v6 = dr("Voti6")
      end while
      'incremento dell'opzione votata
      Select Case opz
        Case "Rb1"
          v1=v1+1
        case "Rb2"
          v2=v2+1
        case "Rb3"
          v3=v3+1
        case "Rb4"
          v4=v4+1
        case "Rb5"
          v5=v5+1
        case "Rb6"
          v6=v6+1
        case else
          exit sub
      End Select
      'update in Sondaggi
      updated = MyUpdateSondaggi(id,v1,v2,v3,v4,v5,v6)
      'inserimento in Votanti
      inserted = MyInsertVotanti(id,ip,oggi)
      'redirect alla pagina dei risultati
       Response.Redirect("risultati.aspx?id=" & id)
   else
       Response.Redirect("risultati.aspx?id=" & id)
   end if
End Sub
```

Alla variabile **id** è assegnato l'ID del sondaggio, ad **opz** il "nome" del **radiobutton** selezionato, **ip** è l'indirizzo IP del visitatore, mentre **oggi** è la data odierna. Il controllo **if… then…else** permette di eseguire il codice solo nel caso in cui non abbia già votato. Per questa verifica ci si avvale della **VotoValido** a cui viene passato l'ID del sondaggio e che restituisce un valore boolean. Il codice necessario, in realtà è solo una leggera modifica della solita "funzione di query" creata da Web Matrix:

```
Function VotoValido(ByVal iD_Sond As Long) As Boolean
 Dim ipAttuale as String = Request.ServerVariables
                                          ("REMOTE_ADDR")
 Dim DataAttuale as DateTime = DateTime.Today()
 Dim connectionString As String = "server='localhost';
             trusted_connection=true; Database='WebPoll'"
 Dim sqlConnection As System.Data.SqlClient.SqlConnection =
       New System.Data.SqlClient.SqlConnection(connectionString)
 Dim temp as boolean = true
 Dim queryString As String = "SELECT [Votanti].[Ip],
      [Votanti].[ID_SOND], [Votanti].[Data] FROM [Votanti] W"& _
                  "HERE ([Votanti].[ID_SOND] = @ID_SOND)"
 Dim sqlCommand As System.Data.SqlClient.SqlCommand =
                        New System.Data.SqlClient.SqlCommand
                                 (queryString, sqlConnection)
```

```
sqlCommand.Parameters.Add("@ID_SOND",
                    System.Data.SqlDbType.Int).Value = iD_SOND
sqlConnection.Open
Dim dr As System.Data.SqlClient.SqlDataReader =
                              sqlCommand.ExecuteReader(
               System.Data.CommandBehavior.CloseConnection)
    while (dr.Read())
        if ((dr("ip")=ipAttuale) And (dr("Data")=DataAttuale)) then
            temp = false
        else
            temp = true
        end if
    end while
return temp
End Function
```

Il voto non è valido, solo nel caso in cui nella tabella **Votanti** sia registrato un utente con lo stesso IP, che abbia già votato a questo sondaggio **(ID_SOND)** nella data odierna. Non è l'unico modo per controllare voti multipli, ma è certamente un compromesso valido, preferibile all'uso di cookies che non sempre sono supportati dalle regole del browser in uso. Nel caso in cui il voto non sia valido, l'istruzione successiva al ramo **Else**, dirige l'utente verso la pagina dei risultati. Se il voto è valido, la pagina recupera dalla tabella **Sondaggi**, mediante il datareader **dr**, il numero di voti contenuti nei campi che vanno da **Voti1** a **Voti6** e li assegna alle variabili da **v1** a **v6**. L'istruzione **SELECT…CASE** successiva, permette di incrementare di un voto il valore della variabile corrispondente alla scelta effettuata. Se l'utente avrà fatto click sulla seconda opzione, la variabile **v2** verrà incrementata. Successivamente viene effettuato un aggiornamento di tale valore incrementato all'interno della tabella **Sondaggi**, mediante il metodo **MyUpdateSondaggi**. **MyInsertVotanti** invece si occupa dell'inserimento nella tabella **Votanti** di una nuova entry, contente l'ID del sondaggio, l'IP del votante e la data odierna. Come ultimo step, abbiamo un redirect verso la pagina dei risultati per mostrare all'utente l'opinione degli altri utenti sommata alla sua scelta. **MyUpdateSondaggi** e **MyInsertVotanti** sono due funzioni create come le funzioni precedenti, ma utilizzando rispettivamente i code builders **"Update Data Method"** e **"Insert Data Method"** e modificate manualmente. Entrambe restituiscono valori interi (assegnati a **updated** e **inserted**) corrispondenti al numero di record (normalmente uno) affetti da cambiamento. Si veda il codice sorgente allegato alla rivista per i dettagli delle due funzioni.

LA PAGINA RISULTATI.ASPX

La pagina **risultati.aspx** deve semplicemente mostrare, in maniera graficamente gradevole, i risultati del sondaggio identificato dall'ID passato come parametro nella barra degli indirizzi. Al caricamento della pagina, la funzione **MyQuery** (identica a **MyQuerySondaggi**), si occupa di fornire un datareader contenete tutte le informazioni relative ai campi della tabella sondaggi aventi l'**ID_SOND** del sondaggio. L'aspetto della pagina è mostrato in **Figura 7**. La routine di

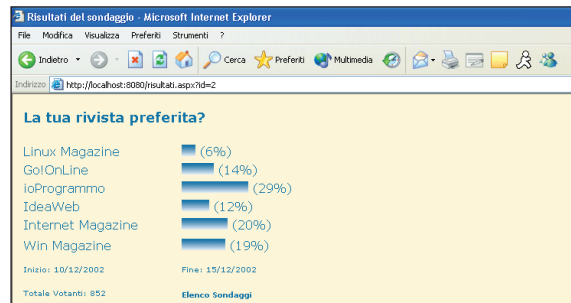



Fig. 7 - La pagina dei risultati di un sondaggio

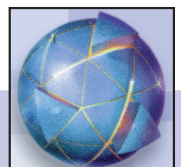
gestione dell'evento caricamento di pagina, assegna a delle variabili i valori raccolti attraverso il datareader: **domanda**, le **sei opzioni**, i **sei voti**, la **DataInzio** e quella di **DataFine.** A questo punto viene calcolata la percentuale dei voti rispetto al totale e assegnata alle variabili che vanno da **v1** a **v6**. Predisposta una tabella nel codice HTML, sarà sufficiente visualizzare i valori delle variabili contenenti la domanda del sondaggio, le varie opzioni e i valori percentuali. Volendo avere una rappresentazione a istogramma bisognerà predisporre un'immagine rappresentante una barra colorata di dimensione fissa, da ridimensionare in maniera proporzionale alla percentuale. Il codice di tutte le sei barre blu accanto alle opzioni, è analogo al seguente:

```
<% Response.Write("<img height=""15"" src=""score.gif""
       width=""" & 300*v1/100 & """/>" & " (" & v1 & "%)")%>
```

300 è la larghezza dell'immagine **score.gif**, mentre **v1** contiene la percentuale di voti dell'opzione1 sul numero totale di voti delle sei opzioni. Se la percentuale fosse 100%, avremmo l'attributo **width = "300*100/100" = "300"**, per cui la barra blu sarebbe visualizzata completamente. Se la percentuale fosse del 50% avremmo la barra visualizzata per metà, e così via. Per motivi di arrotondamento nel calcolo della percentuale, potrebbe accadere che la somma delle sei percentuali, non dia luogo a un 100% esatto, ma sia affetta da un errore di 1%. Per evitare questo si potrebbe porre maggiore cura nell'approssimazione, controllando se è per difetto o per eccesso, oppure più semplicemente dedurre la percentuale della sesta opzione (per esempio) sottraendo da 100 il totale delle altre cinque.

CONCLUSIONI

Web Matrix, attraverso i suoi wizard e l'ottima integrazione con MSDE, ha mostrato ancora una volta le potenzialità nel realizzare in brevissimo tempo un'applicazione ASP.NET. Un miglioramento possibile è quello di rendere l'applicazione più flessibile e gestire un numero variabile di opzioni anziché un massimo di sei. Risulta certamente una buona scelta anche l'aggiunta di controlli circa possibili errori dovuti a ID di sondaggio inesistenti o a problemi di connessione con il server. Un'ulteriore ampliamento interessante è quello di inserire un pannello di amministrazione per l'inserimento e l'editing di sondaggi all'interno di un'area riservata del sito.

Antonio Cangiano

Introduzione a ColdFusion MX

Un giorno di parecchi anni fa, rientrai a casa con un pacchetto tra le mani. Entrai veloce nella mia stanza e, con un'impazienza indicibile, aprii quella scatola, che conteneva l'oggetto che avevo tanto desiderato nei mesi precedenti… il mio primo modem.

SUL CD

\soft\codice
\coldfusion.zip

Ricordo l'entusiasmo che provai quando, una volta collegato l'apparecchio, installati i driver e configurato l'accesso ad internet, feci il numero del mio provider che, dopo un'attesa che sembrava non finire mai, mi lasciò entrare in quello che allora era un mondo misterioso: Internet. Quelli erano i primi anni del world wide web, anni in cui la grande Rete era vista ancora con diffidenza e curiosità, anni in cui se dicevi di possedere un indirizzo e-mail ti guardavano come se avessi eseguito a mente una divisione a dieci cifre. Poi vennero gli anni dell'entusiasmo, della new-economy, l'era in cui si pensava che con Internet fosse possibile fare tutto, trovare di tutto e, soprattutto, vendere di tutto, anni in cui anche il macellaio sotto casa accarezzava l'idea di un suo sito Internet, così da vendere la carne on-line alla stessa vecchietta che ogni sera andava personalmente a comperarla. E poi, piano piano, siamo arrivati ai giorni nostri, periodo in cui la Rete si è oramai dichiarata per quello che è, un'enorme contenitore di informazioni e di opportunità a disposizione di chi voglia utilizzarlo, pronto a mutare con il cambiare delle mode e delle esigenze e capace unificare razze, religioni e nazionalità in un'unica popolazione globale. Sarà forse che lavorando nel settore informatico mi scontro quotidianamente con Internet, ma mi riesce difficile immaginare al giorno d'oggi un computer isolato, senza possibilità di accesso alla grande Rete. Internet è entrata nelle nostre vite prepotentemente, e questa veloce divulgazione del mezzo ha portato ad uno studio costante di nuove tecnologie che permettessero di usufruirne meglio e più velocemente, semplificando lo sviluppo di contenuti distribuibili in Rete ed aumentandone nel contempo la potenza. Nel corso di questo articolo verrete introdotti ad uno di questi strumenti, apparso da poco nella sua ultima versione. Un server web, che rappresenta forse il più semplice modo di sviluppare contenuti dinamici per Internet e che però, a dispetto della semplicità d'uso, propone caratteristiche interessanti per poter essere utilizzato per sviluppare applicazioni on-line altamente professionali. ColdFusionMX fa parte di una famiglia di prodotti Macromedia appositamente studiati per essere integrati in progetti web-based. Ma vediamo di capirci qualcosa di più.

I SERVER WEB

Mi perdoneranno i più colti di voi, che gia sanno quali sono i compiti che un web-server svolge, ma mi preme introdurre l'argomento per tutti coloro che invece non hanno ben presente di cosa si tratti. Abbiamo detto che il prodotto in questione, ColdFusionMX è un server web. Ma cosa è un server web? Un server web è quel programma, residente sul computer server, che una volta digitato un indirizzo nel browser, vi restituisce la pagina web che state cercando. Vediamo di spiegare meglio il tutto con un esempio. Ogni volta che provate a caricare una pagina di un sito che vi interessa digitando una stringa di testo nella barra degli indirizzi del vostro browser, altro non fate che indicare al vostro programma client (il browser) di collegarsi ad un computer server (quello del sito che volete visitare) ad una specifica applicazione che risiede su quel computer(il server web) comunicandogli, tramite un linguaggio di scambio di nome HTTP, che volete visualizzare una determinata pagina di quel sito. Il server web residente risponde quindi al client inviando al browser la pagina richiesta dall'utente sempre mediante protocollo http. Questo nel caso più semplice. È possibile, infatti, che il server web debba compiere delle azioni prima di poter inviare la pagina al client. Prendete ad esempio il caso in cui la pagina richiesta contenga l'esito di una ricerca eseguita tramite uno dei tanti motori di ricerca disponibili in internet. Ovviamente la pagina in questione, anche se è sempre la stessa, dovrà contenere risultati differenti a seconda della parola per cui si è effettuata la ricerca. E chi, secondo voi, deve occuparsi di modificare i contenuti della suddetta pagina ogni qual volta essa deve essere visualizzata in risposta ad una nuova ricerca? Bravi, avete risposto bene, il server web. Infatti tutti i server web moderni, anche se in modi completamente differenti, danno la possibilità di interpretare linguaggi di scripting server-side in modo da rendere la pagina web dinamica nel contenuto. Quindi, in un esempio più aderente alla realtà, quando un client invia una richiesta http al server web, gli spedisce anche dei valori che indicano cosa esso desidera visualizzare tra i tanti possibili contenuti di quella pagina. Il server web riceve quei valori, li controlla e costruisce una pagina web personalizzata a seconda delle richieste ricevute dal browser. L'utente non vede tutto questo processo, egli riceve esclusivamente la pagina che gli interessa e non si pone il problema di come quei dati siano arrivati sul proprio computer. Dal punto di vista del programmatore che deve fare in modo che ciò accada, la faccenda è piuttosto complessa. Il primo tra i tanti problemi da affrontare quando si inizia a sviluppare un'applicazione web a contenuto dinamico è: che tecnologia utilizzerò? Oggigiorno ne esistono molte, ma le più conosciute sono soprattutto tre:

- La piattaforma .NET di Microsoft, programmabile per applicazioni web con il suo linguaggio ASP.NET ed utilizzabile con il server web Internet Information Services sempre di Microsoft.

- La piattaforma Java, programmabile con il linguaggio JSP e tramite blocchi di codice Java chiamati Servlet, ed utilizzabile con il server web Tomcat disponibile gratuitamente.

- La piattaforma MX, programmabile con un linguaggio a tag chiamato CFML ed utilizzabile con il server web ColdFusionMX di Macromedia.

Tutte e tre le piattaforme hanno pregi e difetti, ma mentre le prime due hanno una curva di apprendimento piuttosto lenta, la terza è stata sviluppata con l'intento di renderla più immediata e intuitiva possibile. È di lei che parleremo nel corso dell'articolo e, imparando a conoscerla, vi accorgerete di come sia possibile, attraverso questo prodotto, arrivare presto e con semplicità, a svolgere compiti che con altre tecnologie richiedono un impegno molto, molto più elevato. La piattaforma Coldfusion è adatta per i neofiti quindi, ma ben si presta anche a chi cerca soluzioni professionali di semplice implementazione poiché, come vedremo, tanta semplicità non inficia affatto la potenza del prodotto. Ma ora basta con le chiacchiere ed andiamo a vedere veramente come poter utilizzare questo gioiellino messo a nostra disposizione.

COLDFUSION MX

Una volta in possesso del prodotto in questione, occorre installarlo sul proprio computer, seguendo una procedura piuttosto semplice che andiamo subito a vedere. Dopo le prime schermate, che richiedono i dati dell'utente e i codici del prodotto, la prima schermata interessante che appare è quella riguardante la scelta del server web da utilizzare per utilizzare il prodotto. A questo punto potreste chiedervi: ma come? Sto installando un server web e mi si chiede quale server web utilizzare? In effetti è così, infatti ColdFusion analizza il sistema in cui viene installato e controlla la presenza di eventuali altri web server già presenti. Nel caso che durante l'installazione vengano trovati altri server web, viene messo l'utente nella condizione di scegliere se continuare ad utilizzarne uno preesistente per analizzare tutte quelle richieste http che non necessitano dell'utilizzo di ColdFusion, nel qual caso il prodotto si limita a processare esclusivamente le richieste web che lo interessano, oppure scegliere ColdFusion per assolvere tutte le richieste http che pervengono al server stesso. In ognuno dei due casi, il server ColdFusion viene installato ed è possibile accedervi collegandosi alla porta 8500 del proprio computer. Nei nostri esempi si utilizzerà esclusivamente il server ColdFusion per processare le pagine che via via realizzeremo. La schermata successiva della procedura di installazione riguarda la directory nella quale si desidera installare il prodotto, ed è seguita da un'altra pagina che invece richiede il tipo di installazione da effettuare. Di norma non è necessario modificare le impostazioni predefinite di queste pagine. Subito dopo queste schermate si arriva alla richiesta di una password come Amministratore e come RDS User. Una volta immesse tali password l'installazione è praticamente ultimata e non resta altro da fare che lasciare il tempo al sistema di configurare il prodotto secondo i parametri scelti. Bene, ora avete ColdFusion installato sul vostro computer, non rimane che iniziale ad utilizzarlo per realizzare qualcosa di funzionante.

LA PRIMA APPLICAZIONE

Come primo esempio, per saggiare la semplicità d'uso e la potenza del prodotto, realizzeremo una pagina web che ci permette di eseguire interrogazioni su un database che andremo a creare. I prodotti concorrenti illustrati prima, offrono tutti questa funzionalità in modo relativamente semplice da implementare, ma sfido chiunque a trovare un modo più immediato di quello implementato in Coldfusion.

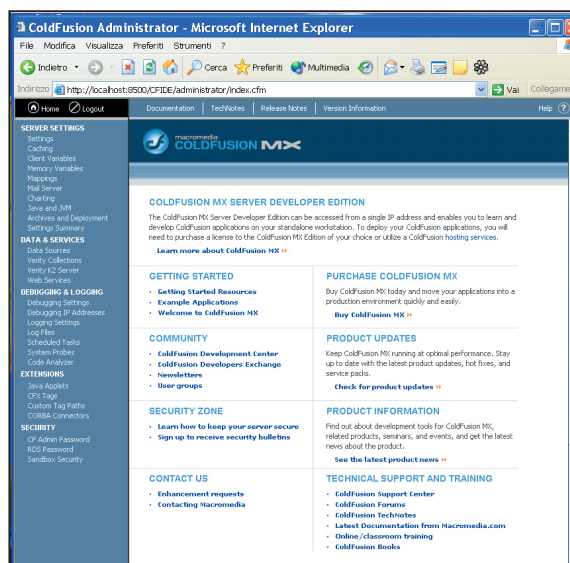
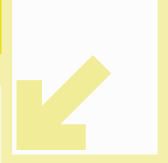



Fig. 1 - Amministrazione di Coldfusion.

Una prima cosa importante da dire riguardo al prodotto trattato, è che esso viene fornito come servizio, quindi non c'è un vero e proprio file eseguibile. In effetti, esso è sempre attivo sul nostro computer, e per accedervi è sufficiente collegarsi alla porta 8500 tramite browser. Vedremo più in là come procedere per utilizzare le funzioni di cui necessitiamo. Ma iniziamo ad interessarci all'esempio.

Prima di tutto occorre creare il database che ci servirà per estrarre i dati di cui abbiamo bisogno: Microsoft Access è più che sufficiente al nostro scopo. Avviate quindi Access e create un Database di nome **"dbProva"** formato di una sola tabella che chiameremo **Persone**. La tabella in questione possiederà tre campi, di nome **nome cognome età**. Se non disponete di Access o non sapete come utilizzarlo, potete prelevare dal CD allegato alla rivista quello relativo all'esempio che stiamo costruendo. Una volta in possesso del database, occorre popolarlo con dei dati fittizi che possano

essere richiamati. Immettete quindi dei nomi, cognomi ed età a piacimento fino a riempire almeno 5 o 6 record. Ovviamente se prelevate il database dal CD, esso è già riempito con dei dati e pronto per essere utilizzato. Per fare in modo che questo database possa essere in qualche modo gestito dalla nostra applicazione web, occorre che il server sia in grado di connettervisi. Occorre quindi mettere al corrente ColdFusion che abbiamo creato il nostro database e che vogliamo utilizzarlo per ricavarne dei dati. Per fare ciò si deve accedere al pannello principale dell'applicazione, in un modo però, che non tutti sono abituati a utilizzare.

Ho spiegato prima che ColdFusion è effettivamente sempre attivo sul nostro computer e per utilizzarlo occorre collegarsi sulla porta 8500 del computer, ma come fare in pratica per utilizzare questa applicazione? Nel menù di avvio della barra degli strumenti, sotto la voce **Start -> Programmi -> Macromedia ColdFusion** appaiono diverse icone che riportano le voci **Administrator, Documentation, Getting Started,** etc.etc. In realtà queste icone altro non sono che collegamenti a pagine web presenti sul proprio PC, che forniscono un'interfaccia con la quale comandare il programma. Proviamo a caricare la pagina di Amministrazione, che poi è quella che utilizzeremo per impostare il collegamento al nostro database. Se osservate nella barra degli indirizzi del browser, che si apre appena cliccato sull'icona, noterete che l'indirizzo della pagina caricata inizia per **http://localhost:8500/**. In effetti è un indirizzo un po' differente dai soliti che siamo abituati a vedere. Le differenze sono date dalla parola localhost e da quel numero che la segue dopo i due punti. Localhost, altri non è che un modo per far capire al nostro browser che il file che cerchiamo non è in internet ma sul nostro computer e, per trovarlo, dobbiamo connetterci alla porta 8500 (il numero dopo i due punti) e seguire tutto il percorso che viene riportato nel resto della stringa. Il connettersi a quella determinata porta, che corrisponde appunto a quella dove è in ascolto ColdFusion, fa in modo che le informazioni passino per il nostro server web, così da rendere possibile la comunicazione con il prodotto. Ma ritorniamo al nostro pannello di amministrazione. Una volta immessa la password, che è la stessa che avete specificato durante l'installazione, sarete liberi di accedere a tutti gli strumenti che vi consentono di gestire tutto il server web, con tutti possibili siti associati.

Ciò che ci serve ora, dicevamo, è permettere ad una nostra applicazione di utilizzare il database che abbiamo realizzato. Per farlo, dobbiamo seguire pochi semplici passi:

- Fate click sul link **Data Sources** presente sotto la voce **DATA & SERVICES** nella colonna di collegamenti sulla sinistra del pannello di amministrazione.

- Nel campo di testo **Add New Data Source** della casella **Add New Data Source**, inserire un nome di riferimento per il database che vogliamo registrare. Sarà grazie a questo nome che potremo poi riferirci al db in questione. Il nome che sceglieremo sarà **Prova**.

- Nella casella combinata sottostante, dove trovate scritto **Driver**, dovete selezionare il tipo di driver da utilizzare per gestire il database in esame. Essendo questo db sviluppato con Microsoft Acces, non dovete far altro che selezionare questa voce dall'elenco.

- Premete ora il tasto **Add** presente alla fine della casella. Verrete trasportati in un'altra pagina dove dovrete indicare dove si trova fisicamente il vostro database.

- Premete il pulsante **Browser Server** corrispondente alla casella di testo **Database File**, verrete trasportati in un'altra pagina che vi permetterà di selezionare il database cercandolo nel sistema.

- Trovate il vostro Database e, dopo averlo selezionato, premete il tasto **Apply** che vi riporterà alla pagina precedente.

- Ora che il file di database è selezionato, non rimane altro da fare che premere sul pulsante **Submit**.

Una volta completata questa serie di passaggi, dovreste vedere nella lista di sorgenti dati disponibili, anche la nuova appena creata con il nome **Prova**. Se nella casella **Status** trovate segnalato **ok**, significa che ColdFusion ha verificato il collegamento e riesce a comunicare con il db. Una volta controllato che il database viene effettivamente riconosciuto dal sistema, altro non rimane che realizzare la pagina che il nostro server deve interpretare e nella quale deve, con criteri che dobbiamo fornire noi, visualizzare la risposta che desideriamo. Per scrivere la nostra applicazione ColdFusion, abbiamo prima di tutto bisogno di un editor di testo. Va bene di tutto, anche il semplice blocco notes di windows, basta avere l'accortezza di salvare il file prodotto con il nome **ProvaCFML** e l'estensione **.cfm**. Una volta aperto l'editor digitate il codice seguente:

```
<html>
<head>
<title>Prova CFML</title>
</head>
<body>
<cfquery name="miaQuery" datasource="Prova">
  SELECT * FROM Persone
</cfquery>
<table border="1" width="100%">
 <tr>
  <td width="100%" align="center" bgcolor="orange">
   Contenuto del database di esempio:
  </td>
 </tr>
 <cfoutput query="miaQuery">
        <tr>
                <td align="center" bgcolor="yellow">
                        #nome# #cognome# #eta#
```

```
                </td>
        </tr>
 </cfoutput>
</table>
</body>
</html>
```

Come potete vedere, il file prodotto altri non è che semplice html con delle aggiunte, sempre però sotto forma di tag, che aggiungono funzionalità al nostro codice. Il linguaggio che ci permette di comunicare con ColdFusion, infatti, viene espresso sotto forma di tag, in tutto e per tutto simili ai normali tag html. Ma andiamo a vedere il funzionamento della pagina nel dettaglio.

COME FUNZIONA COLDFUSION

Il listato di esempio inizia con le normali dichiarazioni che identificano l'inizio, la testa e il corpo del documento html, e prosegue con il primo vero tag CFML. Identificare i tag CFML è semplice in quanto iniziano tutti con i medesimi caratteri **<cf**. Questo specifico tag, di nome **cfquery**, non fa altro che ordinare a ColdFusion di effettuare appunto una query su un database. I parametri che seguono, **name** e **datasource**, indicano rispettivamente il nome da dare alla query e la sorgente di dati dalla quale attingere per ricavare i dati richiesti. Il nome della query serve per poter richiamare in seguito il risultato dell'interrogazione al database e lo utilizzeremo quando dovremo visualizzare il contenuto del db. Il nome della sorgente di dati, invece, altri non è che il nome con il quale abbiamo chiamato il nostro database dal pannello di amministrazione di ColdFusion, cioè **Prova**. Tra l'apertura e chiusura del tag **cfquery** è presente la vera e propria interrogazione al database, sotto forma di stringa SQL. Sarà qui che potrete immettere le vostre richieste da effettuare al db. Il codice prosegue dopo questa prima istruzione CFML con delle dichiarazioni HTML per la creazione di una tabella che ordini i risultati ottenuti dall'interrogazione. Più sotto si arriva alla seconda ed ultima istruzione CFML di questo esempio, che visualizza il risultato della query eseguita prima. Il nome del tag utilizzato in questo caso è **cfoutput**, ed a questo viene associato un solo parametro, di nome query, al quale viene assegnato lo stesso nome della query eseguita in precedenza.

In questo modo si informa ColdFusion che si è interessati a stampare il risultato di quella query piuttosto che altro. Questo ci torna utile poiché l'interrogazione a db prima creata, contiene diversi record e qui, in un'unica istruzione, diciamo al nostro server web di scorrere tutto l'elenco di record presenti nel risultato della query e, per ognuno di questi, richiamare il codice contenuto tra l'apertura e la chiusura del tag cfoutput. E cosa fa il codice presente nel tag? Crea delle nuove righe della tabella ed in ognuna di esse posizione nome, cognome ed età contenute in uno dei record trovati. Semplice no?

Non resta che richiudere tutti i tag HTML aperti in precedenza ed il gioco è fatto. Ecco eseguito in solo due righe di codice effettivo ciò che in altri prodotti comporta una conoscenza ed un impegno decisamente maggiori. Ora che avete capito come il codice della pagina .cfm raggiunge il suo scopo, non rimane che caricarla per fare in modo che diventi effettivamente utilizzabile.

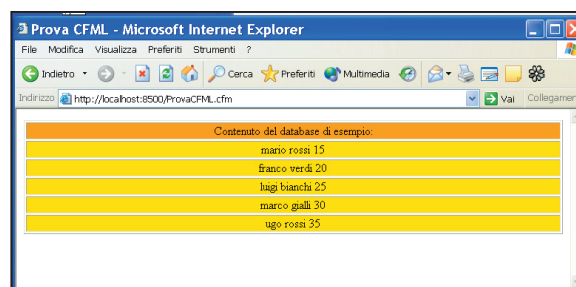



Fig. 2 - Un primo esempio di CFML.

Sfortunatamente, se lanciate la pagina semplicemente cliccandovi sopra non otterrete l'effetto desiderato. È necessario infatti posizionarla in un punto specifico del disco per fare in modo che ColdFusion la trovi e possa utilizzarla. Se ricordate quando abbiamo caricato il pannello di amministrazione, l'indirizzo sulla barra del browser iniziava per **http://localhost:8500/**, e vi ho detto che il nostro server web lo utilizzava come inizio per seguire poi il percorso indicato successivamente nella stringa. Ma effettivamente dov'è l'inizio di questo percorso? La cartella alla quale ColdFusion fa riferimento se noi scriviamo **http://localhost:8500/** è **wwwRoot**, presente all'interno della cartella di installazione di ColdFusion stesso, quindi significa che questo è il punto che possiamo utilizzare per posizionare il file **.cfm** che abbiamo realizzato.

In questo modo potremo caricare la nostra pagina digitando nella barra del browser: **http://localhost:8500/Prova.cfm**. Facendolo, si viene portati direttamente davanti all'interfaccia utente che abbiamo realizzato per la nostra applicazione. Provate pure a cambiare i parametri di ricerca modificando la stringa SQL della query creata per fare in modo che le persone visualizzate siano sempre differenti, e divertitevi con la vostra prima pagina dinamica realizzata in ColdFusion.

CONCLUSIONI

Difficilmente la prima volta che si introduce un web server si porta come esempio la connessione a un database, perché di solito questo argomento è piuttosto ostico da capire per i neofiti. Il fatto che questo articolo introduttivo verta invece su questo argomento la dice lunga sull'intuitività e potenzialità del prodotto, che ha come unico vero neo il fatto di essere un prodotto a pagamento, a differenza di IIS e Tomcat, ed oltretutto di non essere neanche alla portata di tutte le tasche.

Di contro, offre soluzioni e semplicità uniche, che imparerete a scoprire col tempo. Questo infatti non è stato che il primo passo... vi stupirete di come ColdFusion permetterà, con pochi semplici comandi, di svolgere operazioni molto complicate anche soltanto da immaginare.

Alla prossima puntata dunque.

Giuliano Uboldi

Creiamo la sezione protetta del nostro Sito

(Seconda Parte)

Nella prima parte di questo articolo abbiamo creato uno script semplice e funzionale che ci consente di proteggere alcune pagine del nostro sito dall'accesso incondizionato ad esse.

In questa seconda e ultima parte della trattazione aggiungeremo diverse funzionalità che renderanno lo script un programma completo e professionale.

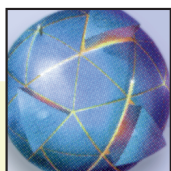
SUL CD
soft\codice
\sezione_protetta_2

L'applicazione che abbiamo imparato a creare durante la lettura del precedente articolo è di per sé perfettamente funzionante, ma chi avrà provato ad implementarla ed utilizzarla si sarà subito reso conto dei suoi limiti e della sua necessità di essere ulteriormente sviluppata. Tale snellezza nella struttura non è stata comunque casuale: in questo modo infatti ho voluto evitare che gli aspetti chiave che il lettore necessitava di focalizzare nella realizzazione di uno strumento di protezione di pagine web venissero confusi con altri aspetti relativi a miglioramenti e "ritocchi", aspetti che vedremo invece in dettaglio in questo articolo.
In sintesi, ciò che ci apprestiamo ad implementare è:

- un sistema di registrazione che consenta a un utente non presente nel database di effettuare la registrazione e poter quindi accedere all'area riservata in un secondo momento;
- la possibilità di memorizzare (all'interno di un cookie) i propri dati durante l'autenticazione, in modo tale che, nei successivi accessi all'area riservata, questi non vengano nuovamente richiesti;
- la possibilità, per un utente che avesse smarrito la sua password, di richiederla e riceverla tramite e-mail (un po' come la funzione presente nella maggior parte dei forum, e non solo, presenti sul web).

AMPLIAMO LA STRUTTURA

Rivediamo ora la struttura dell'applicazione. I file .asp già presenti subiranno alcune modifiche e, in aggiunta, avremo altre tre pagine .asp necessarie per l'implementazione della registrazione dei nuovi utenti e per l'invio della password "dimenticata" tramite e-mail. La pagina iniziale, **loginform.asp**, visualizza il form per l'inserimento di e-mail e password per l'autenticazione sul database. Qui aggiungeremo il controllo sull'eventuale presenza di tali dati all'interno di cookie specifici; in tal caso non verrà visualizzato il form, ma i dati letti dai cookie verranno direttamente spediti a **login.asp** che li utilizzerà per l'autenticazione. Inoltre, inseriremo anche alcune righe che elimineranno i cookie sovracitati nel caso che il loro contenuto non trovi corrispondenze nella tabella utenti del database. Aggiungeremo inoltre due link: uno per l'accesso alla pagina di registrazione nuovi utenti, uno per l'apertura della pagina di richiesta password eventualmente dimenticata. Infine, sempre in questa pagina, avremo un checkbox tramite cui l'utente potrà indicare all'applicazione il suo desiderio di essere "ricordato" nei prossimi accessi, in modo tale che possa evitare di dover reinserire sempre i propri dati. Come indicato nella prima parte di questo articolo, se l'autenticazione in **login.asp** ha esito positivo, l'utente viene redirezionato a **riservato.asp** che è la pagina la cui visibilità sarà effettivamente consentita ai soli utenti registrati.
Veniamo ora alle nuove pagine: **registerform.asp** costituisce il primo step nella procedura di registrazione di un nuovo utente. Essa visualizza un form relativamente simile a quello per il login, con due campi input per l'inserimento, rispettivamente, di email personale e password che si desidera utilizzare nei successivi accessi e un bottone che permette di inviare il form alla pagina **register.asp** che valida le informazioni inviate e, in caso di successo, effettua l'inserimento del nuovo utente all'interno del database.
Il database dell'applicazione è praticamente l'unico elemento dell'applicazione a rimanere perfettamente invariato e, brevemente, esso è costituito da una sola tabella con i seguenti campi:

- **ID:** di tipo intero, contatore; sarà la chiave primaria

della tabella.

- **email:** di tipo testo, conterrà l'indirizzo email di ogni utente.
- **password:** di tipo testo, conterrà la password relativa ad ogni utente.

LOGINFORM.ASP

Come detto, da questa pagina inizierà la procedura di autenticazione tramite l'inserimento di email e password e successivo invio del form a login.asp. In questa sede analizzeremo in dettaglio solo le novità apportate e, precisamente, il controllo sulla presenza o meno dei cookie, il checkbox per l'opzione "ricorda utente" e i link per l'accesso alla pagina di registrazione e di invio password in e-mail.
Vediamo il listato:

```
<%
'se si sono verificati errori nell'accesso elimina eventuali cookie
                                          e abbandona la sessione
if len(request.querystring("err"))>0 then
  response.cookies("user_email") = ""
  response.cookies("user_password") = ""
  session.abandon()
end if
%>
<%
'email e password sono già memorizzati in cookie?
if len(request.cookies("user_email"))>0 then
  'rileva i dati contenuti nei cookie
  dim email, password
  email = request.cookies("user_email")
  password = request.cookies("user_password")
  response.redirect("login.asp?email="&email&"&password=
                                                "&password)
end if
%>
<html>
<head>
  <title> LoginForm - Sezione Protetta </title>
</head>
<body>
<table border=0 bgcolor=#006060 cellspacing=1 width=150
          cellpadding=0 style="font-size:10px; color:#000000;
                                       font-family:Verdana">
  <tr>
    <td align=center bgcolor=#B4C4D3 height=20><b>
                                       LOGIN FORM</b></td>
  </tr>
  <tr>
    <td bgcolor=#E2E8E9>
    <table border=0 cellspacing=0 cellpadding=0 width=100%
      style="font-size:10px; color:#000000; font-family:Verdana">
      <form method=post action=login.asp>
      <tr>
        <td> EMAIL:<br> <input type=text name=
                   email class=input3D size=18 value=""></td>
      </tr>
      <tr>
        <td> PASSWORD:<br> <input type=
                  password name=password size=18 value=""></td>
      </tr>
      <tr>
      ...
    </table>
    </td>
  </tr>
  </form>
</table>
</body>
</html>
```

Nella prima parte della pagina viene fatto un controllo sull'eventuale presenza della variabile **err** all'interno della querystring: se presente significa che si è stati redirezionati al form di login dopo che si è verificato un errore nella pagina **login.asp** dove viene effettuata l'autenticazione vera e propria; in tal caso vengono eliminati eventuali cookie di memorizzazione dei dati utente (vedremo in seguito cosa contengono questi cookie), viene abbandonata la sessione (eliminazione di tutte le variabili ed oggetti di sessione) e viene semplicemente visualizzato il form per l'inserimento dei dati di login. Insomma, un vero e proprio reset dell'applicazione.
Il secondo controllo di questa pagina riguarda la (già citata) presenza o meno dei cookie di memorizzazione di email e password sul pc dell'utente. Per fare ciò si controlla la lunghezza del cookie **user_email:** se uguale a zero, significa che il cookie non è presente, se maggiore di zero il cookie è effettivamente presente.
Nel primo caso viene visualizzato il form per il login, come di norma; nel secondo caso, invece, i dati presenti nei cookie relativi a email e password vengono letti e memorizzati in variabili temporanee e poi direttamente inviati a **login.asp** tramite la querystring; il redirect viene effettuato in questo frangente per cui, in questo caso, non verrà visualizzato alcun form di login, ma verrà effettuata subito l'autenticazione sul database utilizzando i dati appena letti dal cookie.
Proseguendo, all'interno della creazione del form, troviamo il già menzionato checkbox che l'utente potrà selezionare per "farsi ricordare" dal sistema e i due link a **forgotpass.asp** e **registerform.asp**.
Nel secondo caso abbiamo utilizzato un normale tag **<a**

href=...> mentre nel primo link abbiamo creato il link in modo tale che esso apra una piccola popup, utilizzando quindi una funzioncina javascript che racchiude la classica **window.open**, funzione javascript che apre una nuova finestra definita dai valori dei parametri che le vengono passati nella chiamata.

LOGIN.ASP

A differenza della pagina **login.asp** dell'applicazione base che abbiamo realizzato il mese scorso, questa nuova versione presenta un paio di novità. Innanzitutto i dati dell'utente non sono più prelevati da un form tramite il metodo **Request.Form** ma, potendo essi essere inviati sia dal form che tramite la querystring (nel caso di cookie memorizzati), viene utilizzato il metodo **Request** che, essendo più "generico", consente di "coprire" i valori delle variabili sia che esse siano inviate tramite un submit di un form sia che esse siano passate tramite la querystring. Come sottolineato più volte nella prima parte dell'articolo, il punto chiave di **login.asp** è la ricerca nel database dei dati inviati dall'utente e, in caso di autenticazione avvenuta con esito positivo, l'istanziamento di una variabile di sessione **session("userlogged")** che servirà come "biglietto da visita" per tutte le pagine riservate del nostro sito. In questo caso, oltre a ciò, verificheremo se l'utente avrà selezionato o meno il checkbox "remember me" per indicare all'applicazione la sua volontà di essere "ricordato" nei suoi prossimi accessi. Per fare ciò, basta verificare che la lunghezza della **request.form("remember")** sia maggiore di uno; in tal caso istanzieremo due cookie, **user_email** e **user_password**, che conterranno rispettivamente le stringhe corrispondenti all'e-mail e alla password dell'utente e a cui assegneremo (ragionevolmente) scadenza pari a trenta giorni a partire dall'istante dell'autenticazione. Infine, se l'autenticazione si concluderà correttamente l'utente verrà redirezionato a **riservato.asp**, la pagina a contenuto protetto del nostro sito. In caso contrario, cioè se l'autenticazione avrà esito negativo, il client verrà reinviato a **loginform.asp** in cui si avrà la visualizzazione del form con messaggio d'errore connessa all'eliminazione degli eventuali dati utente memorizzati all'interno dei cookie. Vediamo il listato completo di **login.asp**:

FEATURE

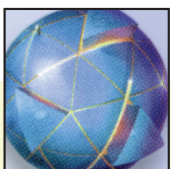
Nuovi feature dell'applicazione:

- **Cookie per memorizzione dati utente.**
- **Registrazione nuovi utenti.**
- **Invio in e-mail della password smarrita.**

```
<%
'rileva i dati inviati dal form o dalla querystring
dim email, password
email = replace(request("email"), "'", "''")
password = replace(request("password"), "'", "''")
dim cn, rs, sql
'connessione al database
set cn = Server.CreateObject("ADODB.Connection")
cn.Open "driver={Microsoft Access Driver (*.mdb)};dbq="
                    & Server.MapPath("sezioneprotetta.mdb")
sql = "SELECT ID, EMAIL, PASSWORD FROM users WHERE
            EMAIL='"&email&"' AND PASSWORD='"&password&"'"
'cerca l'utente nel database
set rs = cn.execute(sql)
'utente trovato
if not rs.eof then
  session("userlogged") = true
  'memorizza dati utente in cookie
  if len(request.form("remember"))>0 then
    response.cookies("user_email") = rs("EMAIL")
    response.cookies("user_email").expires = dateadd("d", 30, now())
    response.cookies("user_password") = rs("PASSWORD")
    response.cookies("user_password").expires =
                                        dateadd("d", 30, now())
...
%>
```

RISERVATO.ASP

Veniamo ora alla pagina protetta dell'applicazione. Rispetto alla versione base implementata nella scorsa lezione l'unica differenza è che, nel caso l'utente decida di uscire dall'ambiente riservato, oltre a venire chiusa la sessione con il comando **Session.Abandon**, vengono anche eliminati gli eventuali cookie di memorizzazione di nome utente e password tramite l'assegnamento ad essi di stringhe nulle.
Eccone il listato:

```
<%
'se l'utente non si era autenticato redirezionalo al form di login
if not session("userlogged") then
  response.redirect("loginform.asp")
end if
'se l'utente precedentemente autenticato vuole uscire
if request.querystring("action")="logout" then
  response.cookies("user_email") = ""
  response.cookies("user_password") = ""
  session.abandon()
  response.redirect("loginform.asp")
end if
%>
<html>
<head>
  <title> Pagina principale - Sezione Protetta</title>
```

```
</head>
<body>
Utente autenticato.<br>
Stai visualizzando una pagina ad accesso riservato.<br><br>
<a href=riservato.asp?action=logout>Clicca qui per uscire.</a>
</body>
</html>
```

REGISTERFORM.ASP

Analizziamo ora la prima delle due pagine dedicate alla registrazione di nuovi utenti (nel database **utenti** appunto). In questa pagina avremo una prima parte di visualizzazione form molto simile al form per l'inserimento dei dati per l'autenticazione, quindi composta da un campo per l'inserimento della propria e-mail e da un campo per l'inserimento della password scelta che si desidera utilizzare in futuro per l'autenticazione. Segue un classico bottone di **submit** che consente di inviare il form a **registerform.asp** (vedremo poi le funzionalità di questa pagina). In coda, viene verificata l'eventuale presenza dell'elemento **err** all'interno della querystring; utilizzeremo tale variabile per passare eventuali messaggi d'errore dalla pagina in cui tenteremo di effettuare la registrazione vera e propria dell'utente all'interno del database. A seconda dell'errore verificatosi essa permetterà di visualizzare tre tipi di messaggi differenti che indicheranno all'utente di quale tipo di problema si sta trattando.
Segue il (piuttosto semplice) listato di **registerform.asp:**

```
<html>
<head>
 <title> RegisterForm - Sezione Protetta</title>
</head>
<body>
<table border=0 bgcolor=#006060 cellspacing=1 width=150
        cellpadding=0 style="font-size:10px; color:#000000;
                        font-family:Verdana">
 <tr>
  <td align=center bgcolor=#B4C4D3 height=20><b>
                        REGISTRATION FORM</b></td>
 </tr>
 ...
<%
 dim errcode
 errcode = request.querystring("err")
 if len(errcode)>0 then
%>
   <tr>
    <td colspan=2 align=center>
    <font color=#ff0000>
<%
  if errcode="email" then
   response.write("Devi inserire un indirizzo email!")
  elseif errcode="password" then
   response.write("Devi scegliere la password che intendi
            utilzzare per l'accesso alla tua area riservata!")
  elseif errcode="mailexist" then
   response.write("Esiste già un utente con lo stesso
                        indirizzo e-mail!")
  end if
%>
    </font>
    </td>
   </tr>
<%
 end if
%>
  </table> </td> </tr> </table> </body> </html>
```

Ecco come dovrebbe apparire il form di registrazione:

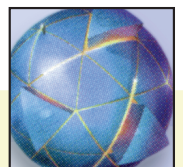
REGISTRATION FORM	
Indirizzo e-mail:	
Password:	
REGISTRAMI >>	

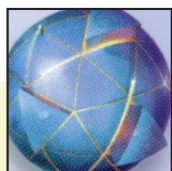
Fig. 1 - Form di registrazione nuovo utente

REGISTER.ASP

In questa pagina avviene la registrazione vera e propria del nuovo utente o, per lo meno, il tentativo di registrazione dell'utente. Lo script si apre con la lettura dei valori di email e password direttamente dal form; segue l'apertura della connessione al database. Prima di effettuare l'inserimento nella tabella controlliamo che l'utente non abbia immesso dati nulli; in caso di valori nulli, a seconda di quale sia l'informazione non inserita, assegneremo la stringa "email" o "password" alla variabile err, variabile che stiamo utilizzando per memorizzare l'eventuale errore da passare a **registerform.asp.** Attenzione, nel caso dell'email, quello che facciamo è semplicemente un controllo sul fatto che sia di lunghezza maggiore di zero (come per la password del resto); un controllo più accurato richiederebbe anche la verifica della correttezza sintattica dell'indirizzo e-mail stesso ma tale trattazione esula dagli scopi di questo articolo; per un'analisi più

COOKIE

**Un cookie è un file temporaneo memorizzato sul PC client e che può essere successivamente letto e/o modificato da altre pagine del "sito" che lo ha creato.
È possibile impostarne la scadenza in modo da poterlo utilizzare in sessioni di navigazione successive.**

approfondita della verifica della validità di un indirizzo e-mail rimando al seguente articolo disponibile in linea in cui sono prese in esame due varianti di tale applicazione, sia con le funzioni ASP tradizionali, sia con l'utilizzo delle Regular Expression: **http://www.aspcode.it/articoli/articoli.asp?act1=show_art&idx=89** Torniamo a noi. A questo punto viene verificato che nel database non sia già presente l'indirizzo e-mail scelto dall'utente; nel caso risulti già registrato, lo script assegna il terzo valore possibile, "mailexist", alla variabile **err**. A questo punto, se non si sono verificati i sovracitati errori, lo script inserisce email e password indicate dall'utente all'interno della tabella **users** del database **sezioneprotetta.mdb** utilizzando una normale query SQL di inserimento (**"INSERT INTO…"**). Nel caso in cui si siano verificati errori, l'utente viene redirezionato a **registerform.asp**, pagina nella quale cui si troverà mostrato nuovamente il form di registrazione con, in aggiunta, il relativo messaggio di errore.

```
<html>
<head>
  <title> Register - Sezione Protetta </title>
</head>
<body>
<%
dim email, password
email = request.form("email")
password = request.form("password")
dim cn, rs, sql
set cn = Server.CreateObject("ADODB.Connection")
cn.Open "driver={Microsoft Access Driver (*.mdb)};dbq="
                        & Server.MapPath("sezioneprotetta.mdb")
dim err
'indirizzo email non immesso?
if len(email)=0 then
  err = "email"
'password non immessa?
elseif len(password)=0 then
  err = "password"
'prova a registrare il nuovo utente
else
  email = replace(email, "'", "''")
  password = replace(password, "'", "''")
  sql = "SELECT ID FROM users WHERE EMAIL='"&email&"'"
  set rs = cn.execute(sql)
  'se non c'è già un utente iscritto con quell'indirizzo email
  if rs.eof then
    sql = "INSERT INTO users(EMAIL, PASSWORD) "&_
        "VALUES('"&email&"', '"&password&"')"
    cn.execute(sql)
%>
- Registrazione avvenuta con successo<br>
- <a href=loginform.asp>Effettua i login >></a>
<%
 'se l'indirizzo email è gia iscritto
 ...
%>
</body>
</html>
```

FORGOTPASS.ASP

Come accennato nella presentazione della struttura dell'applicazione, questa pagina ha la funzione di inviare via e-mail la password agli utenti che la dovessero avere malcapitatamente perduta. La pagina ne racchiude in sé due: se essa viene richiamata senza alcun valore all'interno della querystring, essa visualizza un form con un unico campo di input che l'utente utilizzerà per inserire il suo indirizzo e-mail, quello utilizzato al momento della registrazione, per intenderci. Il form viene inviato alla pagina stessa ma con un valore di variabile all'interno della querystring; ecco l'url associato all'action del form: **forgotpass.asp?act1= send.**
Ed ecco l'aspetto del form per la richiesta della password:

Fig. 2 - Form di inserimento indirizzo e-mail per la richiesta della password personale.

Quando la pagina viene richiamata con **act1=send**, viene eseguita la seconda porzione del codice in essa contenuto e precisamente l'invio in e-mail dei propri dati all'utente che ne ha appena fatto richiesta. Il primo step è la ricerca dell'indirizzo e-mail all'interno del database; in caso di mancata corrispondenza viene visualizzato un messaggio di errore con l'invito a riprovare correggendo l'indirizzo immesso. Se la ricerca ha esito positivo vengono eseguite le righe di codice successive che, tramite l'oggetto **CDONTS** (comunissimo oggetto per l'invio di messaggi e-mail tramite script lato server), spediscono nome utente e password all'utente che le ha richieste, accompagnate da un gentile messaggio di saluto. In questa sede non ci soffermeremo sui dettagli dell'invio

di un messaggio e-mail tramite ASP, argomento che tratteremo in futuri articoli e a cui potrete comunque intanto approcciare tramite un breve script raggiungibile al seguente url: **http://www.aspcode.it/articoli/articoli.asp?act1=show_art&idx=2.**
Di seguito è riportato il listato completo di **forgotpass.asp:**

```
<html>
<head>
  <title> Ricorda Password </title>
  <link rel="stylesheet" type="text/css" href="../aspstyle.css">
</head>
<body>
<%
'se visualizzi il form
if len(request.querystring("act1"))=0 then
%>
  <div style="FONT-FAMILY: Tahoma, Arial; FONT-SIZE: 8pt; COL-
OR: #000000;" align=justify>
  Se hai dimenticato la password per l'accesso all'area utenti,
      inserisci <b>l'indirizzo e-mail che hai utilizzato al momento
          dell'iscrizione</b>, clicca sul pulsante INVIA. Entro pochi
                      secondi ti verrà inviata la tua password.
  <form action=forgotpass.asp?act1=send method=post>
    <div align=center>
    E-MAIL  <input name=email size=20><br><br>
    <input type=submit value="   I N V I A   ">
    </div>
  </form>
  </div>
<%
'invia password in email
elseif request.querystring("act1")="send" then
    'invia via email, la password, all'utente che l'ha dimenticata
  dim email
  email = replace(request("email"), "'", "''")
  dim cn, rs, sql
  'connessione al database
  set cn = Server.CreateObject("ADODB.Connection")
  cn.Open "driver={Microsoft Access Driver (*.mdb)};dbq="
                        & Server.MapPath("sezioneprotetta.mdb")
  sql = "SELECT EMAIL, PASSWORD FROM users WHERE
                                      EMAIL='"&email&"'"
  set rs = cn.execute(sql)
  if rs.eof then
%>
<div align=center><font face=arial size=2 color=
          #ff0000><b>Questo indirizzo non è iscritto all'area
                  riservata.<br><br><br></b></font>
<a href="javascript:history.back();"><font face=arial size=2
                          color=#000000><b>&lt; &lt;
                        Torna indietro</b></font></a>
</div>
<%
  else
    'invia l'email con la password
      dim mail
    set mail = Server.CreateObject("CDONTS.NewMail")
     mail.From = "info@miosito.com"
    mail.To = rs("EMAIL")
    mail.Subject = "Dati area utente"
    mail.Body = "Gentile utente,"&vbcrlf&_
            "ecco i dati per l'accesso alla tua area
riservata:"&vbcrlf&vbcrlf&_
            "E-Mail: "&rs("EMAIL")&vbcrlf&_
            "Password: "&rs("PASSWORD")&vbcrlf&vbcrlf&_
            "A presto."&vbcrlf&vbcrlf&_
            "Lo staff"
    mail.Importance = 2
    mail.Send
    set mail = nothing
    'fine invio email con password
%>
<div align=center><font face=arial size=2 color=#ff0000><b>
  Ti è stata appena inviata una email contenente i dati per
                l'accesso all'area riservata all'indirizzo
             <u><%=rs("EMAIL")%></u><br><br><br>
</b></font>
<a href="javascript:window.close();"><font face=arial size=2
            color=#000000><b>[ Chiudi questa finestra ]
                                    </b></font></a>
</div>
<%
  end if
  rs.close
  set rs = nothing
  cn.close
  set cn = nothing
end if
%>
</body>
</html>
```

CONCLUSIONI E SVILUPPI

In questa seconda parte relativa alla realizzazione di un'area ad accesso riservato possiamo finalmente dire di aver completato e reso molto più funzionale l'applicazione di cui, nel precedente lavoro, avevamo solo creato l'ossatura.
Il progetto qui svolto potrà essere adattato ed utilizzato come base per tutte le sezioni di siti web in cui si renda necessario creare un "confine" ad accesso consentito solamente a un limitato gruppo di utenti.

Paolo Capitani

Cartoline in PHP

In questo articolo vedremo come realizzare in PHP, tramite l'ausilio della libreria grafica GD, un semplice servizio per l'invio di cartoline virtuali.

Cartoline: reali o virtuali? Molti di noi, magari durante qualche spensierata navigazione tra i siti più "allegri" del web, avranno avuto modo di utilizzare uno dei tanti servizi per l'invio di cartoline virtuali. No vi è ancora capitato? Uhm... strano! Comunque vediamo insieme di cosa si tratta...

SUL CD
soft\codice\Postcard.zip

Tutti conosciamo il "principio di funzionamento" della cartoline illustrate; sono quei simpatici gadget fondamentalmente composti da una fotografia o un'illustrazione, inviati, tramite posta tradizionale, da chi si trova in viaggio in giro per il mondo ai rispettivi parenti e amici. Il loro scopo principale è quello di inviare un ricordo ad una persona cara, con tanto di dedica e firma da parte del "viaggiatore".
In realtà oggi ogni occasione è buona per inviare delle cartoline: compleanni, ricorrenze, eventi, ecc. Quindi appare ovvio come un simile sistema di comunicazione sia stato "clonato" anche nel mondo telematico, in modo da poter scambiare non più cartoline reali, composte cioè da carta, e francobolli, ma dei messaggi virtuali.
Come anticipato, in giro per la rete è facile imbattersi in numerosi siti che offrono servizi di questo genere, permettendo di inviare cartoline virtuali ai nostri amici. Quando le cartoline virtuali fecero la loro prima apparizione su Internet erano semplicemente composte da un'immagine e un testo che il "mittente" intendeva recapitare al "destinatario", il quale riceveva tramite email un messaggio di notifica che lo avvisava della presenza di una cartolina a lui indirizzata, con il relativo link da seguire per poterla visualizzare. La cartolina in questione era formata da una semplice pagina HTML composta, naturalmente, dall'immagine scelta e dal testo inviato; i servizi più evoluti facevano scegliere anche il tipo di carattere da utilizzare, il colore di sfondo, la disposizione del testo e così via. Con lo sviluppo delle tecnologie legate al web, soprattutto grazie all'introduzione delle animazioni in Flash, anche le cartoline virtuali hanno cambiato forma, trasformandosi in veri e propri documenti multimediali con tanto di musica ed animazioni.

PROGETTIAMO L'APPLICAZIONE

In questo articolo realizzeremo un semplice sistema per la creazione e l'invio di cartoline virtuali di tipo "tradizionale", cioè composte solo da immagini e testo, ma un po' diverso da quelli che siamo abituati a vedere. Nei sistemi classici, l'invio di una "cartolina virtuale" comporta la memorizzazione, sul server che ospita il servizio, dei dati (immagine e testo) impostati dal mittente, ed il conseguente invio, al destinatario, di un'email di notifica. La cartolina, realizzata tramite codice HTML, verrà creata nel momento in cui il destinatario seguirà il link indicato nell'email di notifica, che lo condurrà al sito che offre il servizio, più precisamente nella pagina che contiene la cartolina a lui destinata. Tuttavia le informazioni necessarie a comporre la cartolina vengono memorizzate sul server che ospita il servizio e qui risiederanno per un periodo di tempo prefissato; se il destinatario non dovesse accorgersi dell'email di notifica, o semplicemente se ne accorgesse troppo tardi, certamente non troverebbe traccia del messaggio a lui inviato. Oppure, cosa succederebbe se il destinatario volesse conservare la cartolina che ha appena ricevuto? Di certo sarebbe un po' scomodo salvare tutti i documenti (sorgente HTML e immagini) che compongono la cartolina. Per risolvere questi piccoli inconvenienti, e per meglio riportare nel mondo "virtuale" il funzionamento delle cartoline del mondo "reale", il nostro sistema funzionerà in modo un po' diverso. Le cartoline che realizzeremo saranno composte da un'unica immagine nella quale verranno "incollati" tutti i componenti della cartolina stessa; in seguito questa sarà inviata direttamente nell'email del destinatario sotto forma di attach. In questo modo il mittente potrà essere certo che la cartolina sarà recapitata al destinatario e quest'ultimo, naturalmente se troverà la cartolina di suo gradimento, potrà conservarla con la stessa semplicità con la quale si copia un file.
Analizzando più in dettaglio la struttura del servizio che intendiamo realizzare possiamo individuare i passi che il mittente dovrà eseguire per inviare la propria cartolina:

- scelta dell'immagine da applicare;
- inserimento dei dati del destinatario e testo della cartolina;
- visualizzazione dell'anteprima ed invio;

Per gestire in modo semplice le fasi di anteprima ed invio della cartolina faremo in modo di memorizzarla, una volta creata, sul filesystem, in una apposita directory, sotto forma di file JPEG. In questo modo tutte le volte che la cartolina dovrà essere utilizzata (ad esempio per essere visualizzata sul browser o per essere spedita) non dovrà essere creata nuovamente. Non dobbiamo infatti dimenticare che la creazione della cartolina comporta la manipolazione di immagini, operazione che, soprattutto se ripetuta più volte per lo stesso cliente, potrebbe incidere negativamente sulle prestazioni del sistema.

STRUMENTI NECESSARI E STRUTTURA DELLA CLASSE

Naturalmente il servizio sarà implementato utilizzando il linguaggio di scripting PHP (**http://www.php.net**) con l'ausilio della libreria grafica GD (**http://www.boutell.com/gd/**), capace di generare e manipolare immagini nel formato JPEG, PNG e WBMP.
Versioni precedenti di questa libreria possono manipolare anche il più diffuso formato GIF, ma dalla 1.6 è stato eliminato a causa dell'algoritmo LZW, utilizzato dal formato GIF per la compressione delle immagini, di proprietà della Unisys. Per poter implementare questo servizio, per prima cosa dobbiamo accertarci che l'installazione del PHP di cui disponiamo, abbia il supporto abilitato per la GD. Per avere queste informazioni ci basterà preparare uno script contenente la chiamata alla funzione **phpinfo()** e successivamente invocarlo dal browser. Tra le informazioni che ci verranno restituite troveremo anche quelle relative alla libreria GD (**Figura 1**).

Fig. 1 - Output della funzione phpinfo()

Altrimenti possiamo attivare il supporto per la libreria GD intervenendo sul file di configurazione del PHP. In particolare, su sistemi Windows dovremo aggiungere (o decommentare) la linea:

```
extension=php_gd2.dll
```

mentre su sistemi Linux, partendo dai sorgenti, dovremo prima configurare il tutto tramite:

```
./configure --with-gd
```

e poi compilare ed installare con i consueti:

```
make; make install
```

A questo punto possiamo dedicarci allo sviluppo vero e proprio dell'applicazione, che sarà realizzata utilizzando la programmazione ad oggetti, in modo da creare un'unica classe (**PW_postcard**) dotata di tutte le funzioni necessarie per la creazione e l'invio delle cartoline. Inoltre, per l'invio vero e proprio della cartolina, faremo uso della classe **smtp**, implementata nel precedente numero della rivista.
I metodi principali di cui disporrà la classe **PW_postcard** saranno **show()** e **send()**; com'è facile intuire il primo servirà a visualizzare sul browser un'anteprima della cartolina da inviare, il secondo invece avrà il compito di contattare il server SMTP e di inviare la cartolina:

```
function show()
{
   if ( !$this->_created )
        $this->_createPostcard();
        echo "<img src=\"".$this->_tmpfile."\">";
}

function send()
{
             if (!$this->_created)
             $this->_createPostcard();
             $client=new smtp();
             $client->from["name"]=$this->mail_from;
             $client->from["email"]=$this->mail_from;
             $client->to=$this->to;
             $client->subject=$this->mail_subject;
             $client->body=$this->mail_text;
             $client->add_attachment(
                         $this->_tmpfile,"image/jpeg","base64");
             $client->connect($this->SMTP,$this->SMTP_PORT);
             $client->sendmail();
             $client->close();
}
```

Naturalmente la classe conterrà altri semplici metodi i cui compiti saranno quelli di fornire all'oggetto tut-

te le informazioni necessarie per la corretta composizione ed invio della cartolina.
Entrambi i metodi **show()** e **send()** faranno uso del **_createPostcard()**, il quale si occuperà della creazione della cartolina attraverso le funzioni messe a disposizione dalla libreria GD e della sua memorizzazione in una directory temporanea, che per default è la **tmp**, che deve essere presente nella stessa directory dello script che utilizza l'oggetto **PW_postcard:**

```
function _createPostcard()
{  $source_size=getimagesize($this->img);
   $src=imagecreatefromjpeg($this->img);
   $n_righe=10;
   $dest_size[0]=$source_size[0]+16;  // nuova larghezza
   $dest_size[1]=$source_size[1]+8+$n_righe*12;
                                              // nuova altezza
   $dest=imagecreatetruecolor($dest_size[0],$dest_size[1]);
   $bgcolor=imagecolorallocate($dest,250,255,215);
   $fgcolor=imagecolorallocate($dest,0,0,0);
   imagefill($dest,1,$source_size[1]+1,$bgcolor);
   imagecopy($dest,$src,8,8,0,0,$source_size[0],
                                          $source_size[1]);
   $logo=imagecreatefromjpeg($this->stamp);
   imagecopy($dest,$logo,$dest_size[
                                       0]-110,12,0,0,97,41);
   $data=date("d/m/Y, H:i");
   imagestring($dest,2,8,$source_size[1]+10,
                                        "Data: ",$fgcolor);
   imagestring($dest,3,40,$source_size[1]
                                       +10,$data,$fgcolor);
   imagestring($dest,2,8,$source_size[1]+22,"Da:",
                                              $fgcolor);
   imagestring($dest,3,40,$source_size[1]+22,
                                   $this->from,$fgcolor);
   imagestring($dest,2,8,$source_size[1]+34,
                                         "Per: ",$fgcolor);
   imagestring($dest,3,40,$source_size[1]+34,
                                     $this->to,$fgcolor);
   $this->text=str_replace("\r","",$this->text);
   $tmp=stripslashes($this->text);
   $tmp=wordwrap($tmp);
   $array=explode("\n",$tmp);
   for ($i=0; $i<sizeof($array); $i++)
         {
            $linea=$array[$i];
            imagestring($dest,2,8,$source_size[1]+50+
                                   (12*$i), $linea,$nero);
         }
         imagestring($dest,2,$dest_size[0]-230,
               $dest_size[1]-18,"In esclusiva per ",$fgcolor);
         imagestring($dest,3,$dest_size[0]-130,
            $dest_size[1]-18,"Programmare il Web",$fgcolor);
         $this->_tmpfile=$this->TMPDIR.$this->
                                       PREFIX.time().".jpg";
         imagejpeg($dest,$this->_tmpfile);
         $fp=fopen($this->_tmpfile,"r");
         $this->image=fread($fp,filesize($this->_tmpfile));
         fclose($fp);
         $this->_created=1;
}
```

Inoltre, per evitare conflitti e sovrapposizioni, i nomi dei file che contengono le cartoline dovranno essere generati in modo tale da risultare univoci. Il metodo che utilizzeremo, semplice ma non del tutto sicuro, consiste nell'utilizzare come nome del file, o solo parte di esso, il valore restituito dalla funzione **time()**. Il numero in questione non è altro che lo Unix timestamp, ossia il numero di secondi trascorsi dalla cosiddetta Unix epoch, vale a dire dalle ore 0:00:00 del 1 gennaio 1970. In questa situazione già creare due cartoline nell'arco dello stesso secondo potrebbe comportare dei problemi; tuttavia per i nostri scopi rappresenta una soluzione accettabile.
Una volta creata la cartolina, dopo essere stata memorizzata nel file immagine in formato JPEG, permarrà nella directory **tmp** fino a quando non verrà invocato il metodo **free()**; questo darà modo all'applicazione, che fa uso dell'oggetto stesso, di utilizzare e visualizzare più volte la cartolina creata.
Per quanto riguarda il metodo **_createPostcard()** è necessario fare una ulteriore precisazione: il metodo deve essere considerato "privato", in quanto non dovrà essere invocato dall'applicazione che fa uso dell'oggetto, ma solo da metodi dell'oggetto stesso.
In realtà questo non può essere garantito dall'interprete PHP che, almeno fino alla recente versione 4.3.0, non supporta ancora il concetto di variabili (o metodi) pubblici e privati, così come avviene in altri linguaggi orientati agli oggetti come C++ o Java.

INVIAMO LE CARTOLINE

La classe **PW_postcard** contiene solo gli strumenti base necessari all'implementazione di un servizio per l'invio di cartoline virtuali, per poter ottenere un servizio funzionante andranno aggiunti altri componenti, come script PHP, pagine HTML di supporto e naturalmente le immagini che raffigurano i soggetti delle cartoline.
La versione presente sul CD allegato alla rivista utilizza il solo script **index.php** per eseguire tutti i passi necessari che portano all'invio della cartolina. In base al contenuto della variabile **$cmd** lo script **index.php**

svolgerà le seguenti funzioni:

- **mostra cartoline (default)**

  la prima pagina mostrata all'utente in modo che possa scegliere la cartolina da inviare. Viene invocata quando non viene passato un parametro **$cmd** valido, oppure se questo dovesse mancare. Utilizza lo script **resize.php** per creare "al volo" le thumbnail delle immagini presenti nella directory specificata dalla variabile **$IMGS_PATH**;

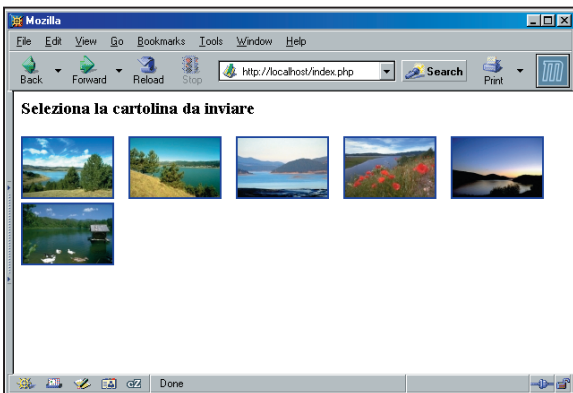



Fig. 2: FASE 1 Scelta della cartolina da inviare.

- **mostra form ($cmd=="form")**

  visualizza la form per l'inserimento dei dati da allegare alla cartolina, e cioè: mittente, email destinatario e testo;

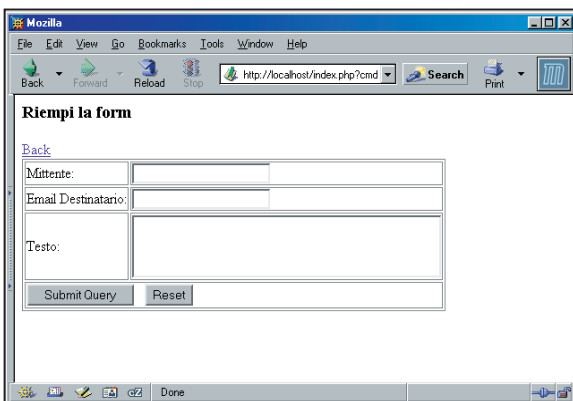
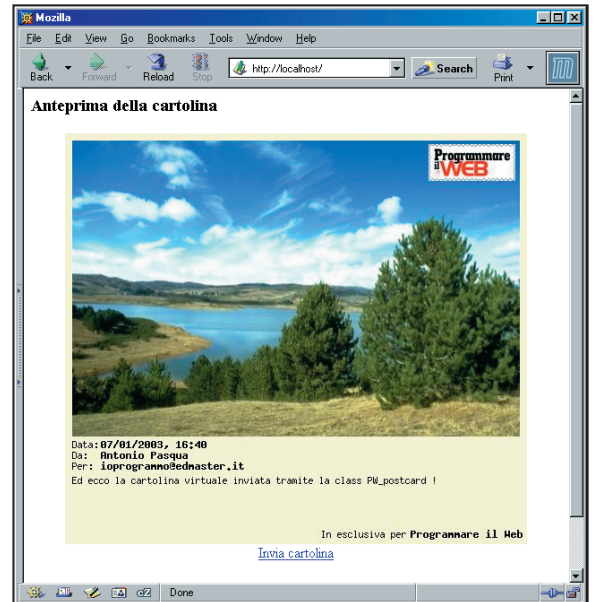



Fig. 3: FASE 2 Inserimento dei dati.

- **mostra anteprima ($cmd=="preview")**

  visualizza l'anteprima della cartolina, in modo da permettere al mittente di verificare come apparirà la cartolina da inviare, ed eventualmente permette di tornare indietro per modificare le scelte fatte;

- **invia ($cmd=="send")**

Fig. 4: FASE 3 Anteprima ed invio della cartolina.

  invia la cartolina al destinatario e la cancella dal disco tramite la chiamata al metodo **free()**;

Da notare, inoltre, che lo script **index.php** fa uso delle sessioni, in modo da memorizzare i dati riempiti dal mittente e riutilizzarli durante le varie fasi della composizione della cartolina. Senza questo meccanismo diventerebbe molto scomodo passare i dati da uno "stato" all'altro dello script, dovendo ricorrere ai classici metodi **GET** o **POST**.

CONCLUSIONI

In questo articolo abbiamo presentato la realizzazione di un semplice servizio per l'invio di cartoline virtuali, utilizzando strumenti gratuiti e facilmente reperibili sulla rete.
Lo scopo dell'articolo era quello di fornire le basi per poter iniziare ad utilizzare le funzioni messe a disposizione dal PHP, e dalla libreria GD, per la manipolazione delle immagini.
L'esempio proposto potrebbe tuttavia costituire la base per l'implementazione di un servizio più completo, ad esempio che supporti la personalizzazione di font e colori, oppure l'utilizzo di un database come sistema di memorizzazione delle cartoline virtuali inviate.

Antonio Pasqua

La sicurezza del software

seconda parte

di Corrado Giustozzi

Nella prima parte di questo articolo, pubblicata lo scorso mese, abbiamo iniziato a parlare dello sviluppo del software, in particolare per quanto riguarda gli aspetti di correttezza e robustezza dei programmi. Abbiamo visto come non sia detto che un programma corretto sia necessariamente anche robusto: ossia, il solo fatto che un programma funzioni correttamente quando tutto va secondo le specifiche non ci assicura che continuerà a farlo quando qualcosa da qualche parte andrà storto. Se poi diamo retta alle Leggi di Murphy dell'informatica, sappiamo che certamente qualcosa prima o poi andrà storto nella vita del nostro programma; quindi non possiamo ignorare le esigenze di robustezza del software che scriviamo. Un programma robusto dunque, oltre ad essere auspicabilmente corretto, cercherà anche di assicurarsi della consistenza della situazione nella quale opera, per accorgersi tempestivamente di eventuali situazioni anomale che dovessero verificarsi e prendere le dovute contromisure: ad esempio cercare di correggere gli errori e ripristinare una situazione di funzionamento nominale o, qualora ciò non fosse possibile, almeno procedere ad una terminazione ordinata e regolare. Fare un programma sufficientemente robusto tuttavia non è facile, anche perché di solito non si è portati a prevedere tutte le possibili cause di malfunzionamento cui il nostro software potrebbe andare soggetto. Di solito, quindi, anche gli sviluppatori di maggiore buona volontà si limitano ad esercitare quel minimo di programmazione difensiva che si fa controllando i codici di ritorno delle principali chiamate di sistema, o disseminando dei costrutti tipo assert nei punti critici del programma. Meglio che niente, certo, ma purtroppo non è abbastanza. Anche perché in determinate situazioni di utilizzo un programma "soltanto" robusto non è sufficiente: serve qualcosa di più, serve che esso sia anche sicuro.

Questo mese vedremo un esempio che ci dimostrerà proprio come anche un programma robusto non è necessariamente sicuro, ossia non protegge contro utilizzi astutamente maliziosi mirati a sconvolgere ad arte il funzionamento del programma stesso.

FORZARE UN CONTROLLO DI AUTENTICAZIONE

Il caso, accettabilmente verosimile pur nella sua comprensibile generalità, riguarda un pezzo di software sicuramente corretto, ed anche ragionevolmente robusto, che si dimostra invece niente affatto sicuro: esso infatti può essere facilmente "ingannato" da una particolare combinazione di dati di ingresso, e portato a produrre risultati non solo errati, ma errati in modo tale da consentire ad un utente malevolo di ottenere illecitamente un beneficio o un vantaggio.

L'esempio è quello, semplicissimo, di una form Web che effettua un controllo di autenticazione allo scopo di limitare l'accesso alle pagine seguenti, lasciando cioè passare solo coloro che dimostrano di essere in possesso di credenziali valide.

Concettualmente questo controllo si fa in un modo elementare: un qualsiasi linguaggio di scripting legato alla form riceve in input le credenziali impostate dall'utente remoto, ossia un codice di login ed una password, e con esse interroga il database locale contenente la lista degli utenti validi e delle relative password. Se la query è soddisfatta, ossia se viene effettivamente trovato nel database un record contenente i due campi indicati, l'utente è abilitato; altrimenti l'utente viene respinto.

Supponiamo dunque che la nostra form di autenticazione abbia due campi che l'utente remoto deve riempire: chiamiamoli UserID e Password. Lo script che gestisce la form, ricevuti i contenuti di questi campi, li userà per costruire una query SQL con cui interrogherà il database server contenente gli utenti autorizzati. Questo database si chiama utenti e contiene due campi chiamati rispettivamente **utente** e **password**.

A livello di form le variabili relative ai due campi di input sono chiamate, ad esempio:

CORRADO GIUSTOZZI

Romano doc, classe 1959. Giornalista scientifico ed esperto di sicurezza informatica. Ha iniziato a scrivere di computer nel 1979, e da allora ha pubblicato oltre mille articoli e tre libri. È stato direttore tecnico e vicedirettore di MCmicrocomputer, ed ha fondato e diretto Byte Italia. Attivo sin dal 1985 sui temi della sicurezza e della crittografia, ha tra l'altro collaborato col Comando Generale dell'Arma dei Carabinieri in indagini sulla criminalità informatica. Attualmente è security evangelist presso Secure Edge di Roma. La cosa più seria che ha fatto è sviluppare per l'Istituto dell'Enciclopedia Italiana il software di giochi linguistici del Vocabolario Italiano Treccani su CD-ROM. Il suo Web è http://www.nightgaunt.org

```
$FORM_UID e $FORM_PWD.
```

Bene, in queste ipotesi assai generali un tipico esempio di codice in grado di generare la suddetta query potrebbe essere di questo tipo:

```
SELECT * FROM utenti
WHERE utente = '$FORM_USR'
AND password = '$FORM_PWD'
```

Così facendo, come si vede, si selezionano dal database utenti tutti quei record in cui il campo utente ha il valore fornito nella variabile **$FORM_UID** e il campo password ha il valore fornito nella variabile **$FORM_PWD**. Il risultato è **TRUE** se la query ritorna una tabella non vuota, ossia se viene trovato (almeno) un record che la soddisfa. Da notare che il record ritornato dalla query non viene affatto controllato: ciò che serve è solo sapere se esso esiste, ossia se la query stessa ha restituito un valore **TRUE** o no.
Per sincerarci che tutto funzioni come si deve, vediamo cosa succede al nostro script in un caso reale pratico. Se l'utente ad esempio immette nel campo **UserID** la stringa **"Mario"** e nel campo **Password** la stringa **"Pippo"** (ovviamente senza le virgolette), la nostra query diventa:

```
SELECT * FROM utenti
WHERE utente = 'Mario'
AND password = 'Pippo'
```

Ciò è esattamente quello che vogliamo. Sembra dunque tutto corretto, e talmente semplice e lineare da non poter nascondere nessun problema, vero? E invece no. Purtroppo questo pezzo di codice è corretto e anche robusto ma non sicuro, perché immettendo nel campo **Password** una stringa **"grimaldello"** appositamente costruita è possibile forzare il controllo di autenticazione ed ottenere l'accesso anche non disponendo di credenziali valide. Vediamo come.
Supponiamo che la stringa inserita nel campo **Password** sia, letteralmente: **"boh' OR TRUE--"** (sempre senza le virgolette) e vediamo cosa succede al nostro script. Dopo la sostituzione, la query diventa:

```
SELECT * FROM utenti
WHERE utente = 'Mario'
AND password = 'boh' OR TRUE--'
```

Notiamo che ora il risultato della sostituzione è un comando SQL caratterizzato da una semantica completamente differente da quella che ci aspettavamo. Adesso infatti la query, grazie alla clausola **OR** artatamente inserita, ritornerà sempre il valore **TRUE**, indipendentemente dal valore impostato per i campi utente e password, e dunque apparirà al chiamante come sempre soddisfatta. Il risultato è che l'utente remoto otterrà il permesso di accesso pur avendo presentato delle credenziali inesistenti. Semplicemente, il controllo è stato ingannato!
Da notare l'uso accorto dei due trattini in fondo alla stringa **"grimaldello"**: essi servono per impedire l'errore di sintassi che verrebbe altrimenti generato per via del numero dispari di apici risultanti nella stringa SQL costruita dallo script. I due trattini sono infatti il simbolo SQL di commento, e fanno sì che tutto ciò che li segue (ossia l'apice "dispari" in fondo alla riga) venga ignorato. Il risultato è un'istruzione SQL sintatticamente corretta e perfettamente lecita.
Perché funziona un trucco del genere? Semplicemente perché il programmatore non è stato... paranoico, ma si è limitato a fidarsi del buon funzionamento "naturale" del programma. In altre parole, egli non ha inserito un controllo di validità sul contenuto dei campi **$FORM_UID e $FORM_PWD** perché ha pensato che non ne valesse la pena. Il suo ragionamento è stato, più o meno, il seguente: se uno o entrambi di tali campi contenessero un valore invalido, la query certamente non troverebbe una corrispondenza valida nel database e ritornerebbe quindi il valore **FALSE**, il che negherebbe comunque l'accesso all'utente. Al programmatore non è venuto in mente che un utente astuto e malizioso avrebbe potuto inserire una stringa fatta in modo tale da modificare il significato della query, facendole tornare sempre il valore **TRUE** a prescindere dal reale contenuto dei campi inseriti, cosa che invece è puntualmente successa.
Come difendersi da un attacco del genere? Ad esempio assicurandosi che i valori dei campi **UserID** e **Password** non contengano apici o trattini (che dunque devono essere banditi come caratteri invalidi dalle reali password degli utenti legittimi), o verificando l'innocuità del comando SQL risultante prima di passarlo effettivamente in esecuzione al database server.

MA C'È DELL'ALTRO...

Purtroppo questo esempio non è l'unico. Utilizzando sempre una stringa **"grimaldello"**, in uno scenario analogo a quello appena visto, è addirittura possibile, sempre sfruttando l'ingenuità del programmatore responsabile della routine di controllo, installare sul server un software a piacimento e mandarlo in esecuzione! Ma di questo, e di altro ancora, parleremo con più agio il prossimo mese.